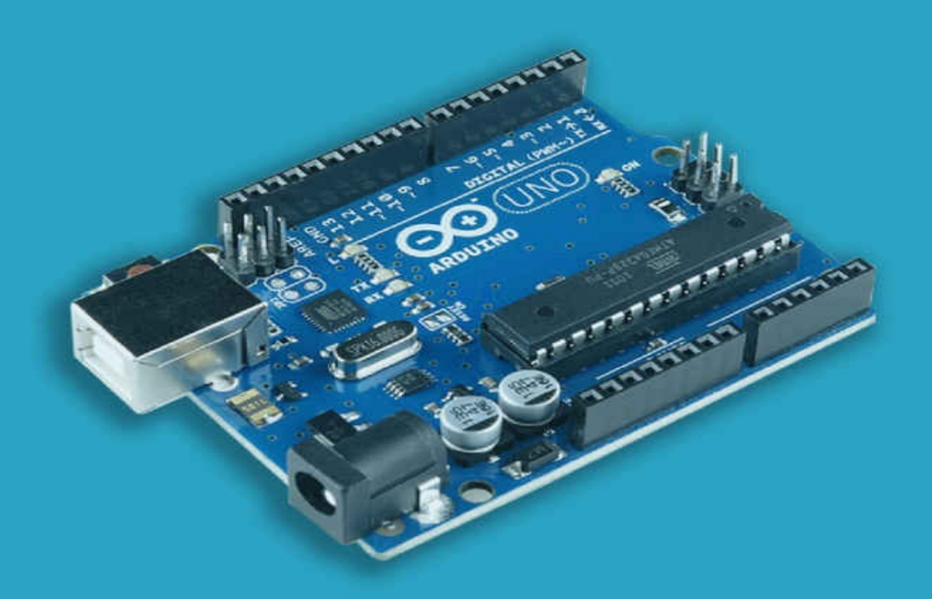


Aprende Arduino en un fin de semana



ALFREDO MORENO MUÑOZ
SHEILA CÓRCOLES CÓRCOLES

El contenido de la obra ha sido desarrollado exclusivamente por los miembros del equipo de **Time of Software**.



Para más información visita:
www.timeofsoftware.com
www.aprendeenunfindesemana.com

¡Bienvenid@! ¡Si estás leyendo ésto es porque has decidido aprender Arduino! ¡Buena idea!

Todo el mundo ha pronunciado o ha escuchado alguna vez en su vida estas dos frases refiriéndose al hecho de aprender algo nuevo:

- No tengo tiempo para aprenderlo.
- Es muy difícil.

Seguro que sí, ¿verdad? Pues con este libro te vas a dar cuenta que aprender Arduino es fácil y… ¡que se puede hacer en un tiempo record!

Hemos diseñado un método con el que podrás aprender todo lo que necesitas saber para tener una base que permita ir desenvolviéndote poco a poco en cualquier proyecto que te plantees hacer con Arduino. El método se basa en un aprendizaje progresivo, adentrándose y profundizando en aquellos conceptos básicos de electrónica y programación que necesitas entender para comenzar a trabajar en el "Mundo Arduino". Todo ello… ¡en un fin de semana!

Una vez hayas acabado el libro, siguiendo el modo de aprendizaje que te proponemos, podemos garantizarte que vas a ser capaz de tener la autonomía suficiente para llevar a cabo tus propios proyectos, o al menos lanzarte a que lo intentes, ya que habrás adquirido unos conocimientos suficientes que te permitirán diseñar y montar tus propios circuitos, desarrollar el código de programación y lo mejor de todo, ver los resultados de tus propios proyectos.

Estamos seguros de que si nos acompañas hasta el final del libro se te van a ocurrir una cantidad grande de ideas para realizar con Arduino, ya que cuantos más conocimientos vas aprendiendo, más curiosidad desarrollarás y más ideas te irán surgiendo.

Te animamos a que comiences a adentrarte en este mundo y disfrutes con cada proyecto. No desesperes si no lo consigues a la primera, ya que seguro que de cada error aprendes algo que te sirve para seguir avanzando. Ésto es solo el comienzo, y Arduino no tiene límites, al igual que tu imaginación.

Para aprender Arduino en un fin de semana tal y como te proponemos aquí, necesitarás una serie de elementos que te permitirán realizar todos los retos que te planteamos.

Lo que necesitas para poder empezar a trabajar con Arduino es:

- Un **ordenador**, con total independencia del sistema operativo que tenga instalado. Si no dispones de conexión a internet deberás de descargar desde cualquier ordenador el ejecutable de instalación del entorno de desarrollo de Arduino e instalarlo en el ordenador que vas a utilizar durante todo el aprendizaje. En los apartados siguientes te explicaremos los pasos a seguir para instalar el entorno de desarrollo en cada uno de los sistemas operativos soportados por la plataforma de desarrollo de Arduino.

- **Placa de Arduino.**

- **Kit de componentes electrónicos básico.**

Tanto la placa como el kit puedes adquirirlo en Amazon.

Y por supuesto… **¡un fin de semana!**

ORGANIZACIÓN

El aprendizaje está dividido en dos partes claramente diferenciadas:

- Teoría y puesta en marcha
- Práctica

La primera parte está compuesta por una fase exclusivamente teórica, en la que aprenderás conceptos básicos para que te desenvuelvas sin problema con Arduino. Además, esta parte incluye todo lo necesario para que seas capaz de montar toda la infraestructura software que necesitas para empezar a trabajar con Arduino, junto con la explicación detallada del entorno de desarrollo.

El aprendizaje práctico está dividido en cuatro Objetivos diferentes y dos Proyectos para afianzar los conocimientos adquiridos en los diferentes Objetivos.

Los **Objetivos** tienen dificultad incremental; a medida que se va avanzando se van adquiriendo nuevos conocimientos de mayor complejidad que los anteriores. Los Objetivos están compuestos por diferentes ejercicios que llamaremos Fases. En cada Objetivo se indica, antes de empezar, los materiales que se necesitan y las funciones de programación comunes a todas las Fases que componen el Objetivo.

Una **Fase** es un ejercicio que profundiza en un área de conocimiento dentro del Objetivo. En cada Fase se indica el montaje físico del circuito y el código del programa Arduino.

Los **Proyectos** son ejercicios de dificultad avanzada que permiten afianzar los conocimientos adquiridos en los Objetivos anteriores. Durante el aprendizaje se realizan dos Proyectos. El primero de ellos está basado en los dos primeros Objetivos, y se realiza después de acabar el segundo Objetivo y el segundo se basa en los cuatro Objetivos y se realiza al acabar el cuarto Objetivo.

DISTRIBUCIÓN DEL FIN DE SEMANA

El método de aprendizaje ha sido diseñado y optimizado para que seas capaz de aprender Arduino en un fin de semana. Obviamente, el tiempo de aprendizaje puede verse modificado ligeramente por los conocimientos previos que tengas.

La secuencia de aprendizaje que debes seguir para alcanzar el objetivo de aprender Arduino es la siguiente:

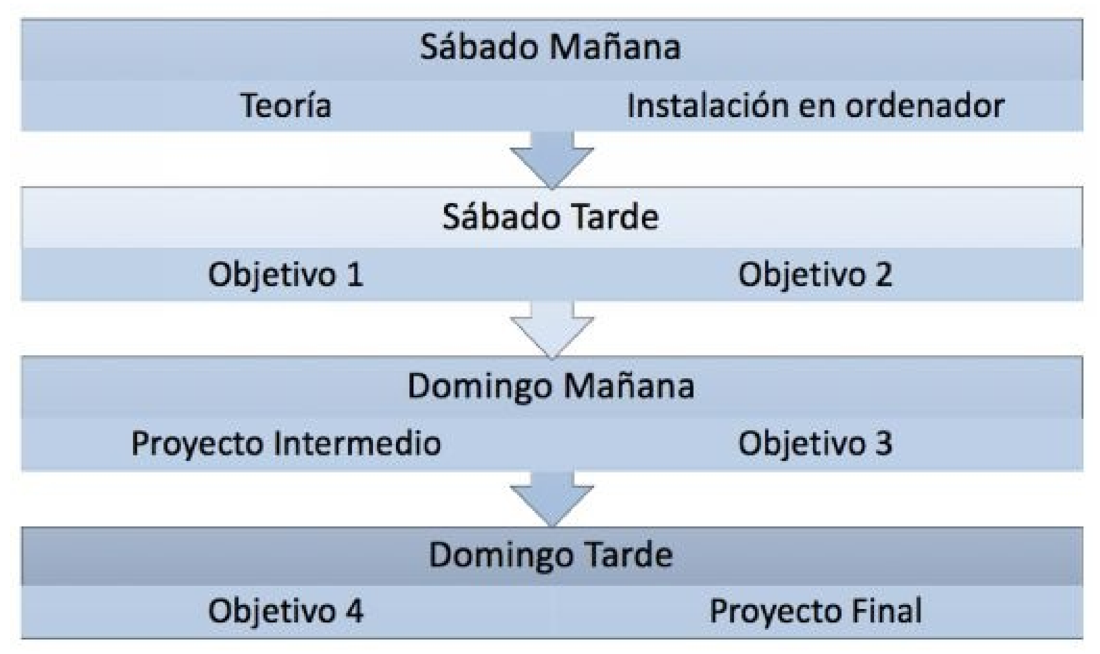


Ilustración 1. Organización del aprendizaje

A continuación te mostramos todos los términos relevantes que vamos a utilizar durante todo el libro y que necesitas conocer para llevar a cabo el aprendizaje.

Código fuente

Es el conjunto de líneas de texto que forman un programa. Las líneas de texto indican cómo se debe ejecutar dicho programa y lo que tiene que hacer.

El código fuente se escribe en un lenguaje específico de programación que tiene que ser traducido al lenguaje que entiende el ordenador.

Sentencia

Una sentencia es cada una de las líneas del código fuente.

Lenguaje de programación

Lenguaje formal utilizado por los ingenieros de software para escribir programas. Mediante el lenguaje de programación se indican todas las sentencias que debe de ejecutar el programa.

Compilar

Proceso de traducir el código fuente al lenguaje que entiende el ordenador.

Lenguaje máquina

Lenguaje que entiende el ordenador y al que es compilado el código fuente.

Bucle

Sentencia específica que se repite durante un tiempo. El número de repeticiones puede ir en función de diversos factores, pero están indicados en la propia sentencia en la que se define el bucle.

Constante

Una constante es un valor que no cambia en toda la ejecución del programa. Para crear constantes en el código fuente se utilizan sentencias específicas.

Variable

Una variable es un valor que cambia durante la ejecución del programa. Para crear variables en el código fuente se utilizan sentencias específicas.

Librería
Conjunto de funcionalidades que se incluyen en los programas y que no son desarrolladas en él ya que pertenecen a otras aplicaciones.

Interfaz
Elemento software que sirve para comunicar dos elementos entre sí mediante el conjunto de operaciones que define.

Pin
Cada una de las entradas o salidas que tiene la placa de Arduino. Un pin puede configurarse como entrada o como salida y es donde conectaremos los componentes para interactuar con ellos. La placa de Arduino tiene diferentes tipos de pines: digitales, analógicos, etc.

Salida
Proceso de enviar datos desde la placa de Arduino a los componentes del circuito electrónico.

Entrada
Proceso de recibir datos desde los componentes del circuito electrónico a la placa de Arduino.

Prototipo
Es el resultado de realizar un proceso de prototipado. El proceso consiste en la construcción rápida del objetivo a conseguir sin tener un conocimiento profundo de todas las fases del proceso gracias a la utilización de técnicas y herramientas que lo permiten.

Hardware
Conjunto de elementos físicos o materiales que constituyen un ordenador o un sistema informático.

Software
Conjunto de programas y procesos que permiten al ordenador la realización de determinadas tareas.

CPU
Conjunto de elementos hardware de un ordenador que interpretan

las instrucciones de un programa informático mediante la realización de las operaciones básicas aritméticas, lógicas y de entrada/salida del sistema.

Procesador
Elemento de la CPU que interpreta las instrucciones y procesa los datos de los programas.

Circuito
Conjunto de componentes que forman una red eléctrica cerrada.

Circuito Integrado
Estructura de pequeñas dimensiones sobre la que se fabrican circuitos electrónicos y que contiene todos los elementos que componen el circuito.

Microprocesador
Procesador de muy pequeñas dimensiones en el que todos los elementos de la CPU están agrupados en un solo circuito integrado.

La Robótica se entiende como la ciencia y la técnica que estudia el diseño y la construcción de robots. Además, se centra en la utilización de los mismos para desempeñar tareas de manera automática o realizar trabajos difíciles o imposibles para los seres humanos.

Un robot es una máquina programable capaz de interactuar con el entorno que le rodea, moverse, mostrar comportamiento inteligente, sentir el entorno, etc.

Uno de los aspectos más importantes de la Robótica es que aúna en una misma ciencia a diferentes ciencias, como son la informática, la electrónica, la ingeniería y la mecánica.

La Robótica es una de las ciencias con mayor auge y que más está avanzando, por ejemplo, se está empezando a utilizar en campos como la medicina para realizar operaciones de dificultad alta, la industria del automóvil, etc.

LEYES DE LA ROBÓTICA

Hablando de Robótica nunca debes dejar de lado las tres leyes descritas por Isaac Asimov en sus novelas. La primera vez que fueron nombradas en sus libros fue en 1942, en el libro *Runaround* y dicen lo siguiente:

1. Un robot no hará daño a un ser humano o, por inacción, permitir que un ser humano sufra daño.
2. Un robot debe hacer o realizar las órdenes dadas por los seres humanos, excepto si estas órdenes entrasen en conflicto con la 1ª Ley.
3. Un robot debe proteger su propia existencia en la medida en que esta protección no entre en conflicto con la 1ª o la 2ª Ley.

Las tres leyes suponen la rendición completa de los robots a los humanos, es decir, asumen el papel de esclavos de los humanos. Veamos:

- En la primera ley se antepone la integridad de la persona ante cualquier otra cosa, es decir, es la protección básica del ser humano respecto a la máquina.

- En la segunda ley se establece la cadena de mando entre el ser humano y el robot. Este último deberá obedecer siempre al ser humano, siempre y cuando no sea para realizar daño a otro ser humano.

- En la tercera ley se establece la rendición absoluta de los robots a los seres humanos. A la vez que los robots siempre deben protegerse para sobrevivir, deberán ser completamente sumisos respecto al ser humano, debiendo autodestruirse si reciben la orden de hacerlo.

En 1985, el propio Isaac Asimov, enunció una nueva ley mediante la cual quedarían todas subordinadas a ella. Esta ley se considera la ley cero, y dice:

- Un robot no puede causar daño a la humanidad o, por inacción, permitir que la humanidad sufra daño.

Las leyes de Asimov se consideran la base para determinar el comportamiento básico que deberían tener todos los robots.

¿POR QUÉ APARECE ARDUINO?

Esta es la primera pregunta que debes de hacerte ya que con ella comprenderás como Arduino ha conseguido el éxito que tiene ahora mismo.

La construcción de circuitos electrónicos no es un proceso sencillo, requiere una variedad de conocimientos en diferentes ramas que hace que no esté al alcance de muchas personas. El proceso se complica aún más cuando los circuitos electrónicos están acompañados de programas software que manejan el comportamiento del propio circuito.

Por tanto, para llevar a cabo el proceso con éxito, la persona o el equipo encargado de llevar a cabo el proyecto, deben tener conocimientos amplios de diversas ramas científicas, lo que hace que nos encontremos con un proceso de construcción de circuitos electrónicos complejo.

En este punto es cuando aparece Arduino, rompiendo todos los moldes sobre los que se trabajaba hasta ese momento, ya que, utilizando la potencia de los microcontroladores o microprocesadores consigue simplificar el uso de la electrónica, pero sin ocultar su dinámica de funcionamiento.

Gracias a Arduino se pueden conseguir resultados de forma rápida y sencilla, poniendo al alcance de todo el mundo la realización de proyectos de robótica mediante la utilización de prototipos.

¿QUÉ ES ARDUINO?

Arduino es una plataforma de código abierto de prototipos electrónicos que se basa en hardware y software flexibles y fáciles de usar que ponen al alcance de cualquier persona la construcción de circuitos electrónicos/robots.

En lo referente a hardware, se basa en placas que se pueden ensamblar a mano o que se pueden comprar directamente preensambladas. Cada una de las placas lleva un microcontrolador en el que se carga el programa software que es necesario desarrollar para "darle vida" a la placa.

En la siguiente imagen puedes ver la Placa Arduino Uno R3 con sus partes más importantes señaladas que serán descritas en el apartado siguiente:

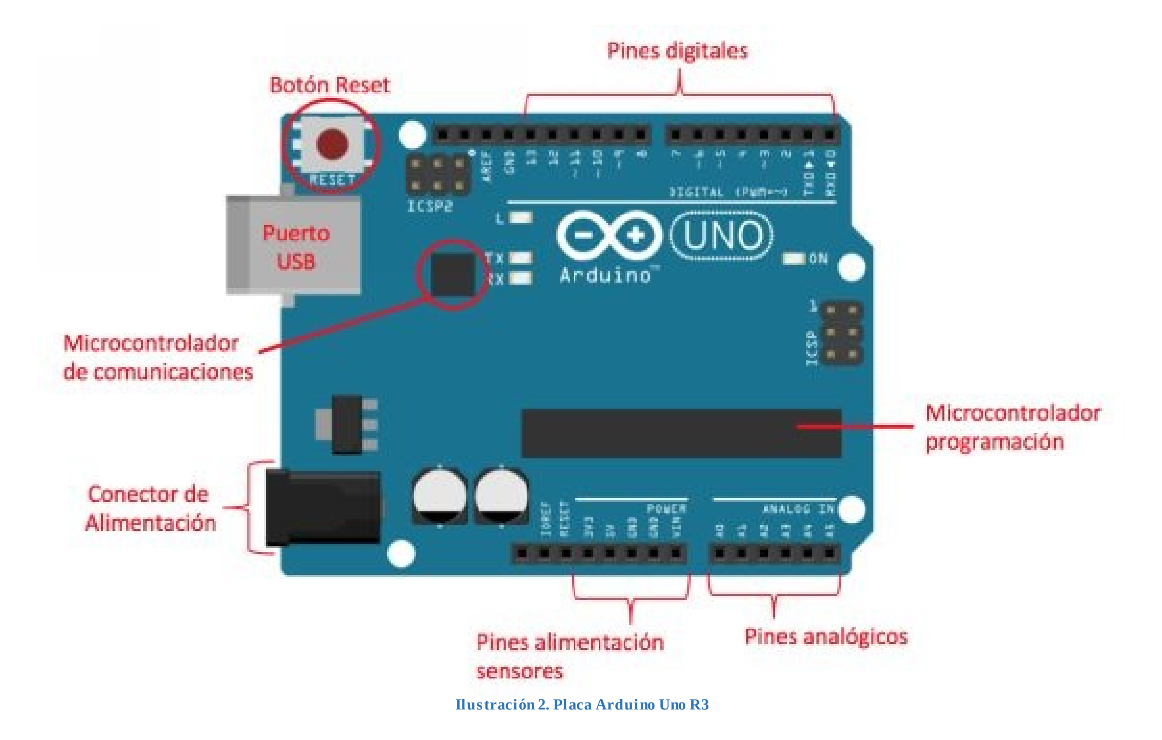


Ilustración 2. Placa Arduino Uno R3

COMPONENTES PLACA DE ARDUINO

PINES DIGITALES

Los pines digitales son las conexiones digitales de los dispositivos conectados en la placa. La placa de Arduino cuenta con 14 pines digitales, que van del 0 al 13.

Una señal digital solo puede tener dos estados:

- 0 (LOW, bajo, false): Indica 0V de tensión enviados desde la placa.
- 1 (HIGH, alto, true): Indica 5V de tensión enviados desde la placa.

Por lo tanto, cuando ponemos un pin digital a valor HIGH, la placa suministra 5V de tensión por la salida que hayamos indicado, y si ponemos el valor a LOW suministrará 0V de tensión. *(Ojo: Hay que tener en cuenta que los 5V no siempre son 5 ni los 0 siempre son 0)*

Los pines digitales de Arduino pueden ser usados tanto de entrada como de salida.

PINES ANALÓGICOS

Los pines analógicos pueden medir un rango de valores de voltaje, a diferencia de los digitales que solo entienden dos valores: 0-1, o lo que es lo mismo, 0V o 5V.

Con los pines analógicos vamos a poder **leer** valores intermedios entre 0V y 5V, representados con un valor entero comprendido entre **0 y 1023**, ya que la información se representa en números de 10 bits, y también vamos a poder **escribir** en los pines valores comprendidos entre **0 y 255**, ya que la información se representa en números de 8 bits.

En el punto anterior hemos hablado sobre pines digitales, si te fijas en ellos verás que aparecen algunos con el símbolo “**~”** en la placa, este símbolo indica que pueden ser utilizados también como pines analógicos.

PINES ALIMENTACIÓN SENSORES

Además de los pines de entrada y salida descritos anteriormente, Arduino dispone de pines que nos permiten alimentar componentes externos, concretamente uno con 5V y otro con 3,3V.

También dispone de pines de tierra (GND).

MICROCONTROLADOR DE COMUNICACIONES

El microcontrolador de comunicaciones se encarga de gestionar las comunicaciones con todo lo que se conecta a la placa.

MICROCONTROLADOR DE PROGRAMACIÓN

Este componente de la placa es el cerebro de la misma, es donde la placa almacena el programa que tiene que ejecutar y el que lo ejecuta.

El microcontrolador de la placa se programa utilizando el IDE (Entorno de Desarrollo Integrado) de programación gratuito de Arduino. En los apartados siguientes explicamos cómo instalarlo y como ponerlo a funcionar.

BOTÓN RESET

El botón Reset permite reiniciar el programa que se ha cargado en el microcontrolador interrumpiendo la ejecución actual. Ten en cuenta que no borra el programa que se ha cargado, únicamente lo **reinicia**.

PUERTO USB

El puerto USB es el puerto mediante el cual nos comunicaremos con la placa de Arduino. Sus funciones principales son:

- Alimentación
- Cargar los programas en el microcontrolador.
- Envío de información desde la placa al ordenador.

CONECTOR DE ALIMENTACIÓN

Arduino dispone de un puerto de alimentación externo que nos permitirá hacer funcionar la placa sin utilizar un ordenador. Tienes que tener en cuenta el no alimentar la placa con mas voltaje del que soporta, ya que podrías dañarla. Lo recomendado es alimentarla entre 7V y 12V.

¿ARDUINO Y ROBÓTICA?

Arduino es considerado una de las bases sobre las que empezar a trabajar en temas relacionados con robótica, ya que, por su simplicidad, permite adquirir conocimientos básicos para dar el paso posteriormente a tecnologías más complejas y completas.

VENTAJAS

Existen multitud de microcontroladores y plataformas en el mercado, pero ninguna de ellas ha conseguido tener tanto éxito como ha tenido Arduino, y eso es debido a que presenta una notable cantidad de ventajas respecto a sus competidores:

- **Precio:** El coste de las placas Arduino es ridículamente barato comparado con el resto de placas. La placa más barata cuesta en torno a 30€ e incluye todo lo necesario para realizar cualquier tipo de proyecto.

- **Multiplataforma:** El software de Arduino se ejecuta en sistemas operativos Windows, Macintosh OSX y GNU/Linux.

- **Facilidad de uso:** El entorno de programación y la placa son lo suficientemente sencillos para que personas principiantes puedan usarlos sin problemas.

- **Software Extensible y flexible:** Arduino no está pensado únicamente para que personas principiantes aprendan, sino que también permite que personas con conocimientos avanzados puedan realizar proyectos complejos. Además, al tratarse de software libre, el código fuente se encuentra accesible y permitirá a usuarios más avanzados acceder y extender las funcionalidades disponibles.

- **Hardware abierto y extensible:** Arduino está basado en microcontroladores que tienen sus módulos publicados bajo licencia *Creative Commons*, lo que permite a diseñadores experimentados poder hacer su propia versión del módulo, extendiéndolo y mejorándolo.

Arduino tiene un entorno de desarrollo propio que te permite cargar en la placa los programas que escribes. Existen dos formas diferentes de utilizar el entorno de desarrollo, el primero es utilizarlo en un navegador web y el segundo es instalártelo en el ordenador.

Para instalarlo entra en https://www.arduino.cc, y después entra en el menú Software:

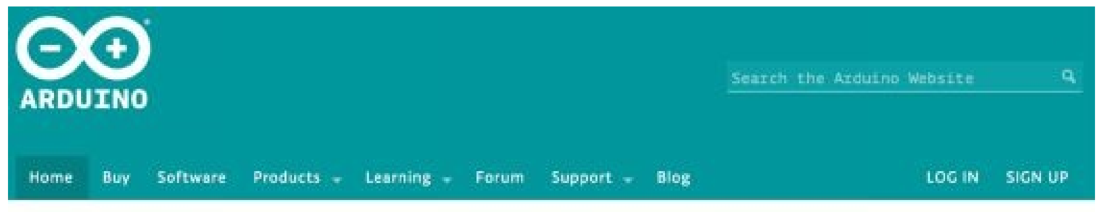


Ilustración 3. Menú principal Web Arduino

También puedes entrar directamente en https://www.arduino.cc/en/Main/Software. En esta página web podrás acceder a cualquiera de las dos formas disponibles del entorno de desarrollo.

A continuación, te explicaré como puedes acceder a cada uno de ellos. Dejo a tu elección el entorno a usar, ya que ambos tipos ofrecen las mismas posibilidades.

ENTORNO WEB

Para utilizar el entorno de desarrollo web tienes que registrarte previamente, por lo que, ése es el primer paso. Para ello tienes que entrar en “*Try it now*” y posteriormente en “*Sign Up*”, rellena toda la información que te piden y como último paso tienes que activar la cuenta mediante el email que te enviará al finalizar el proceso.

Ilustración 4. Arduino Web

Una vez has acabado el proceso de registro tienes que entrar utilizando el *login* y la *contraseña* que has introducido durante el proceso de registro. Al entrar la primera vez tienes que aceptar las condiciones de uso:

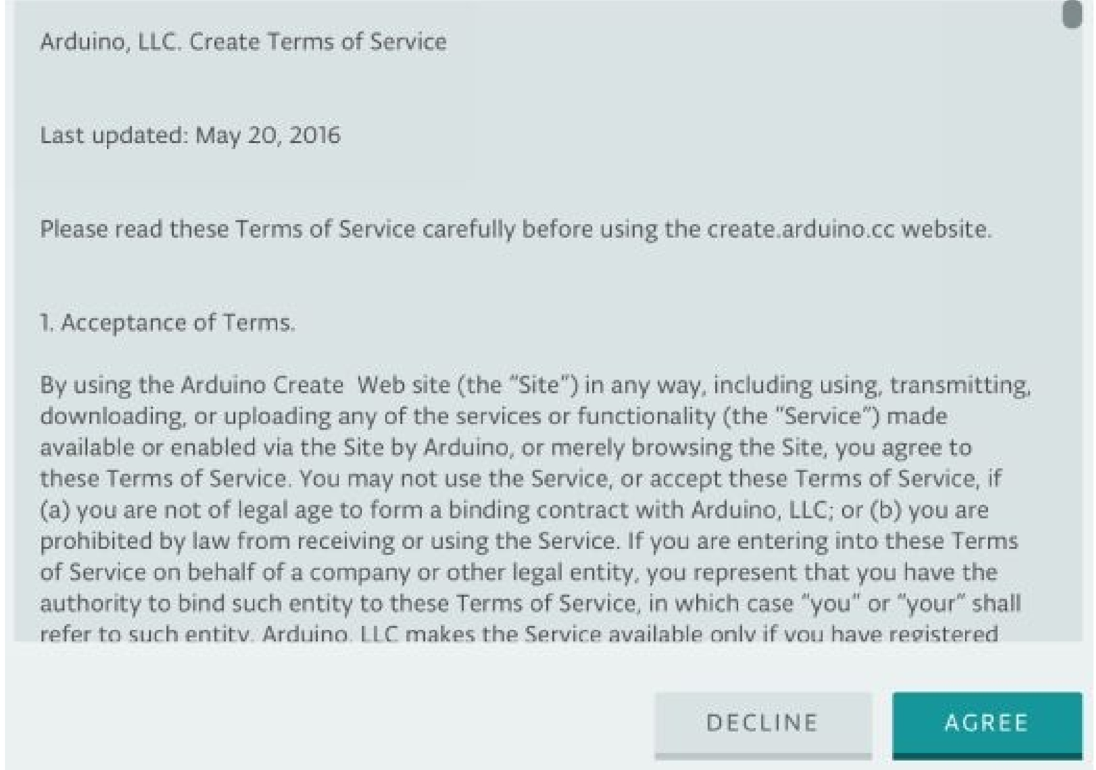


Ilustración 5. Plugin Arduino

Una vez aceptas las condiciones tienes que permitir que Arduino instale el plugin necesario para el sistema operativo que tienes:

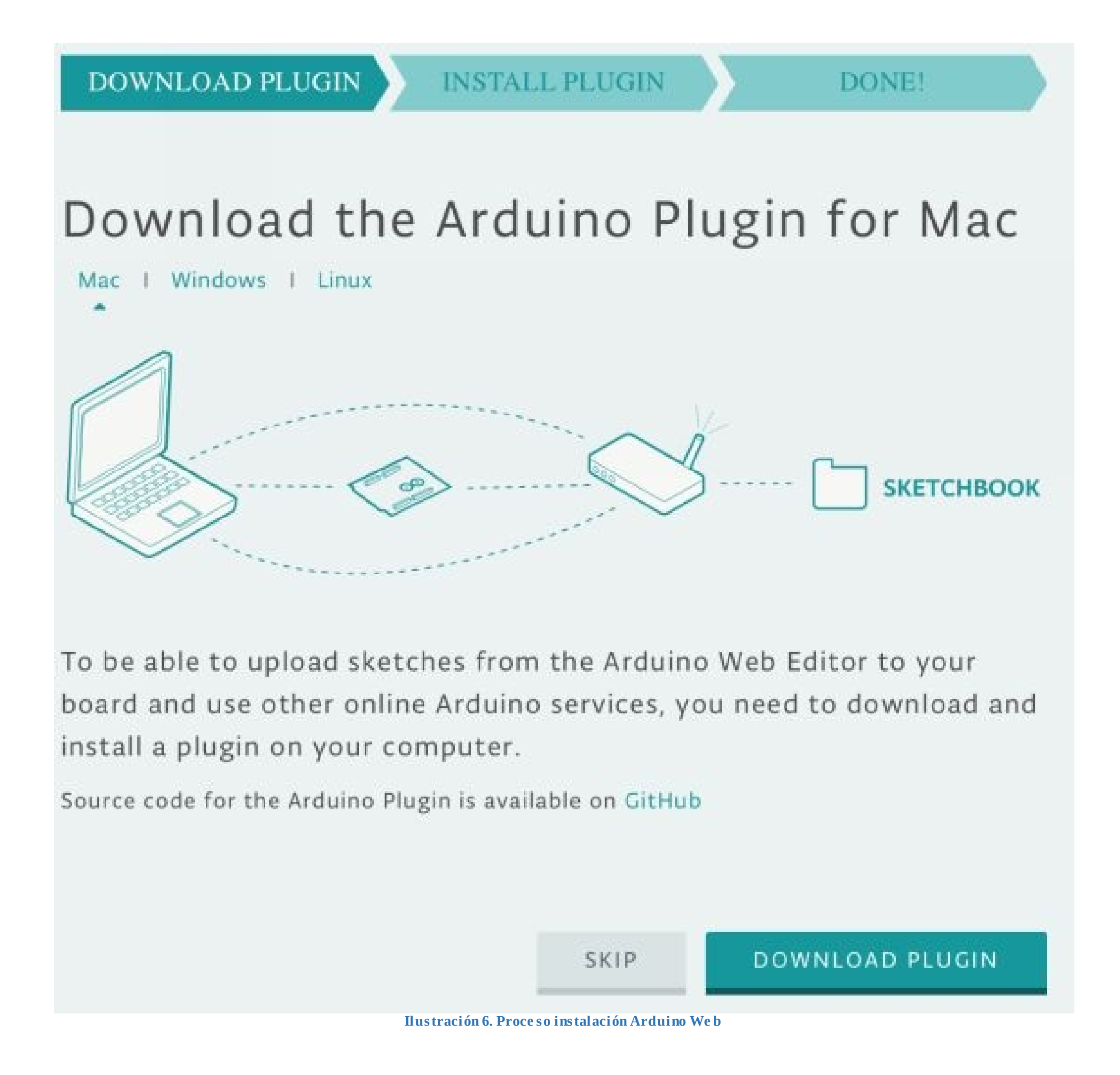


Ilustración 6. Proceso instalación Arduino Web

Elige el sistema operativo que tienes y sigue las instrucciones de instalación, tendrás que aceptar las condiciones y proceder con la instalación. Una vez tengas el plugin instalado, podrás acceder a *Arduino Create*:

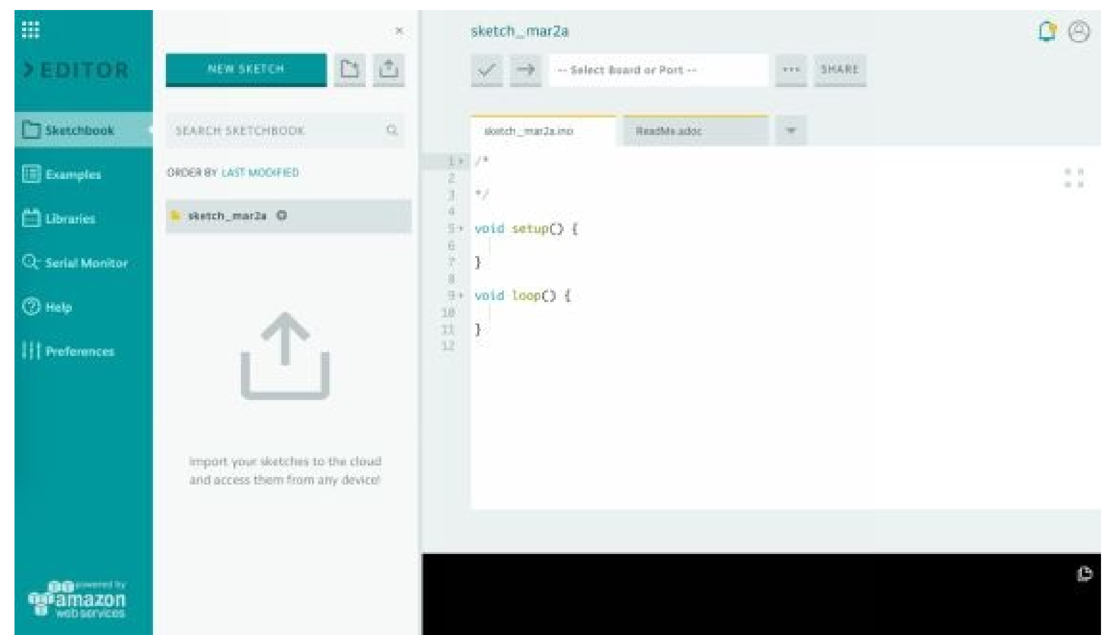


Ilustración 7. Arduino Web

APLICATIVO

En la sección en la que puedes descargar el entorno de desarrollo tendrás la posibilidad de descargarlo para el sistema operativo que estés utilizando:

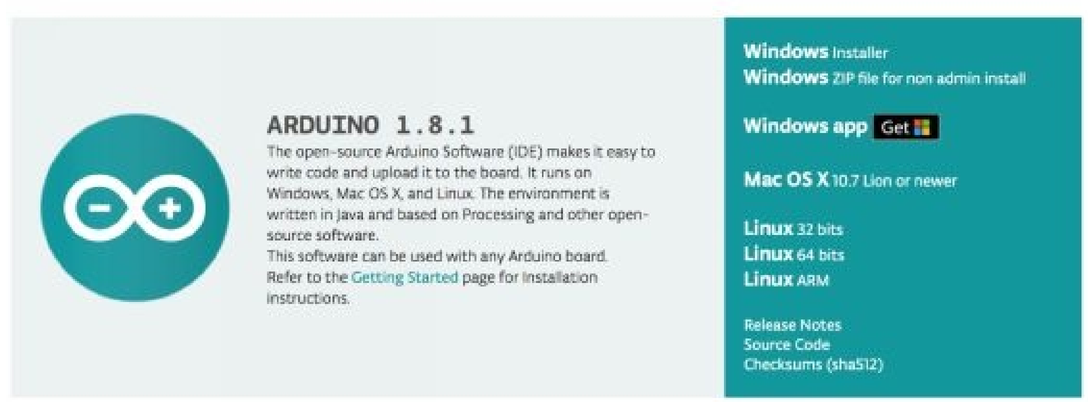


Ilustración 8. Descargar aplicación para Arduino

Selecciona la versión que quieres descargar y te llevará a la página de descarga, desde la que podrás hacer algún donativo para Arduino o directamente descargarlo:

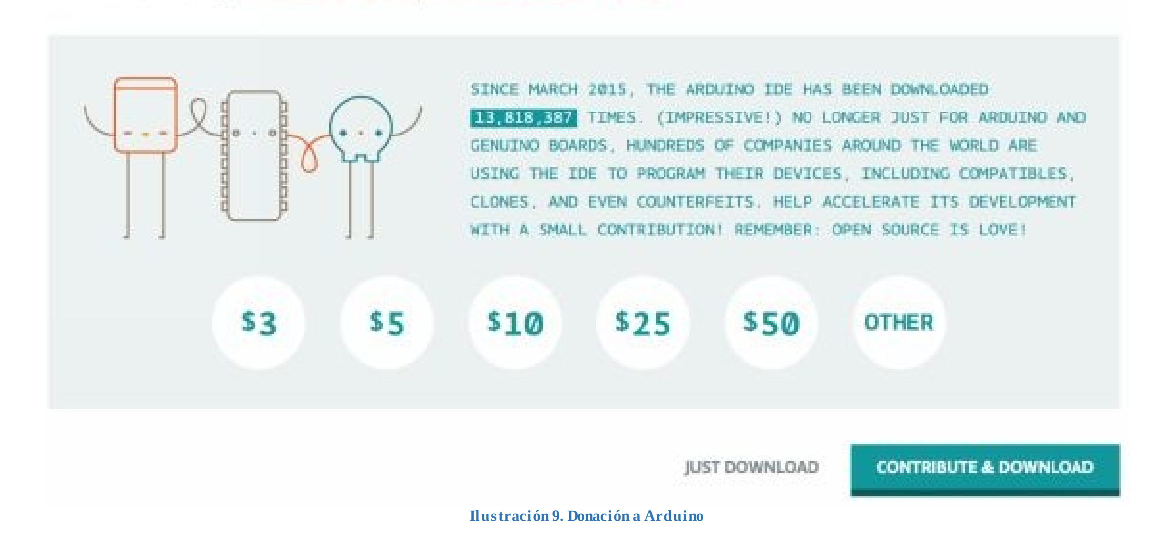


Ilustración 9. Donación a Arduino

INSTALACIÓN DE ARDUINO EN MACOS

Para instalar la versión de macOS únicamente tienes que copiar el fichero de descarga a la carpeta de las aplicaciones:

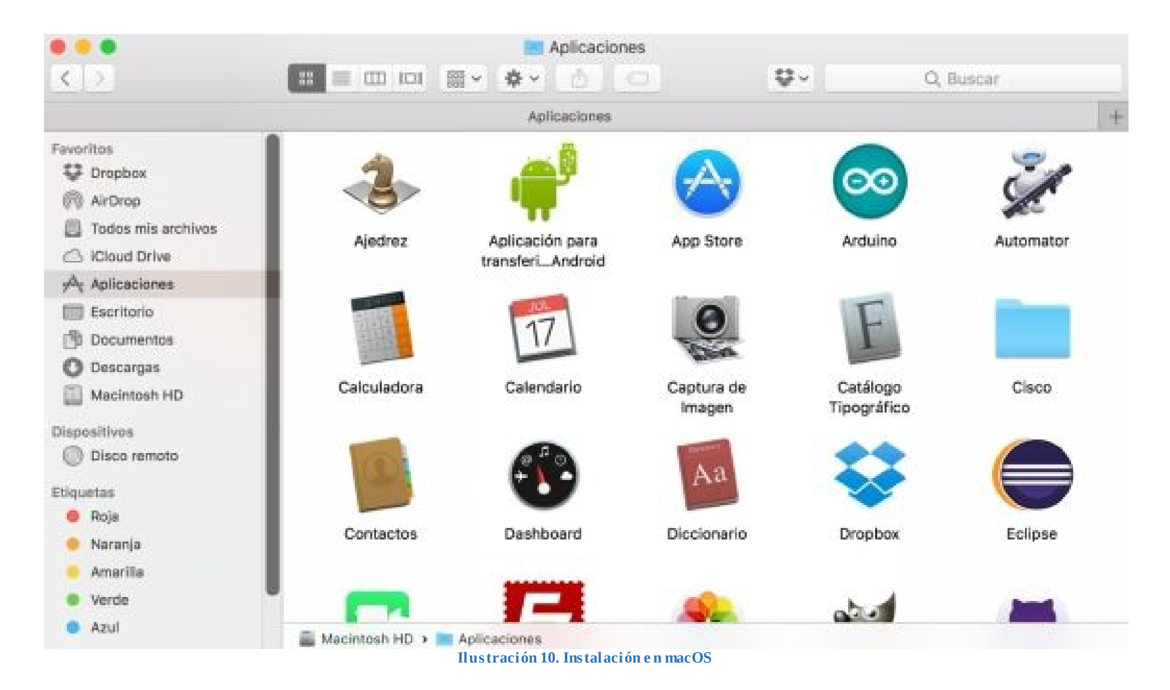


Ilustración 10. Instalación en macOS

Puedes copiarlo a cualquier carpeta, pero te recomendamos que lo copies a la carpeta donde están instaladas todas las aplicaciones y así accedas desde el Launchpad.

INSTALACIÓN DE ARDUINO EN WINDOWS

Para instalar la versión de Windows ejecuta el instalador descargado y acepta las condiciones que te muestra la primera pantalla del instalador:



Ilustración 11. Instalando en Windows (Paso 1)

Una vez aceptadas las condiciones tienes que indicarle qué componentes quieres instalar:

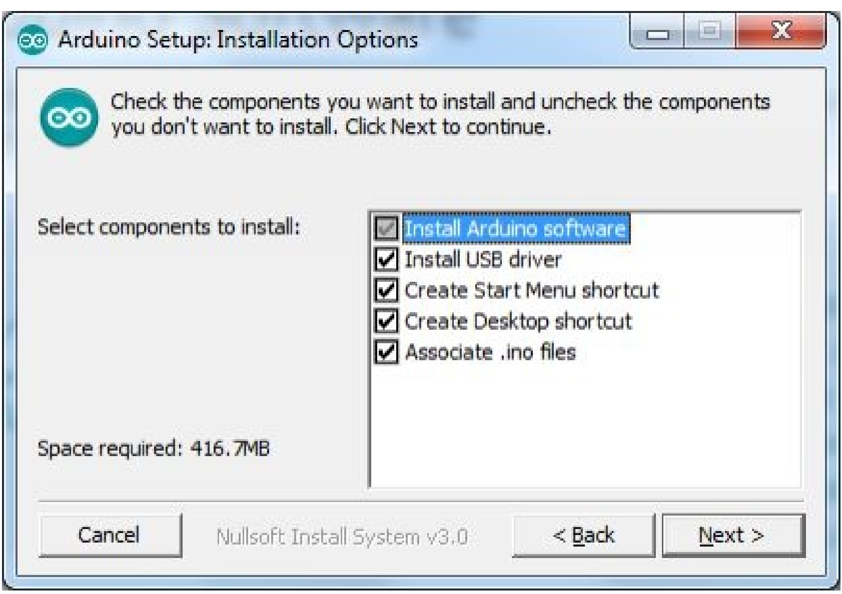


Ilustración 12. Instalando en Windows (Paso 2)

Tras seleccionar los componentes a instalar tienes que indicar la ruta donde quieres instalar el entorno de desarrollo:

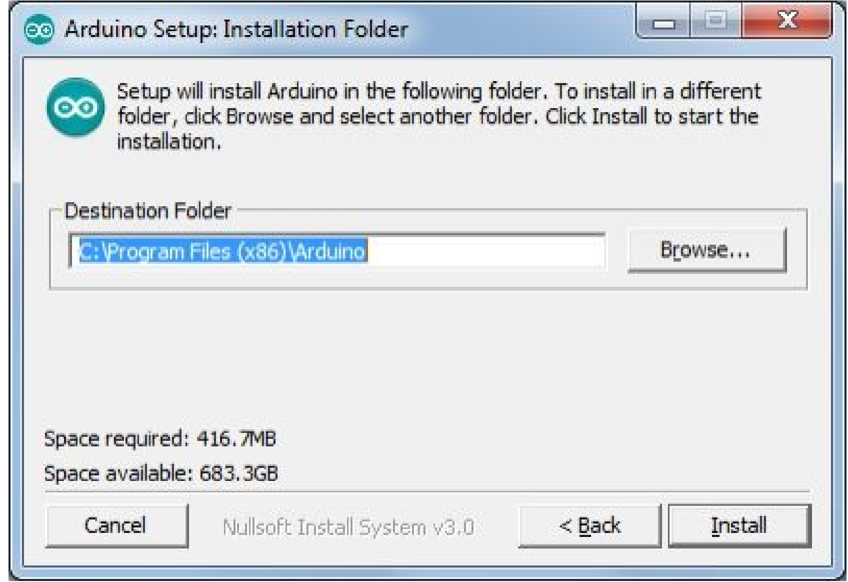


Ilustración 13. Instalando en Windows (Paso 3)

El proceso de instalación empezará una vez selecciones la ruta, una vez acabe cierra el instalador y ya tendrás el entorno listo para empezar:

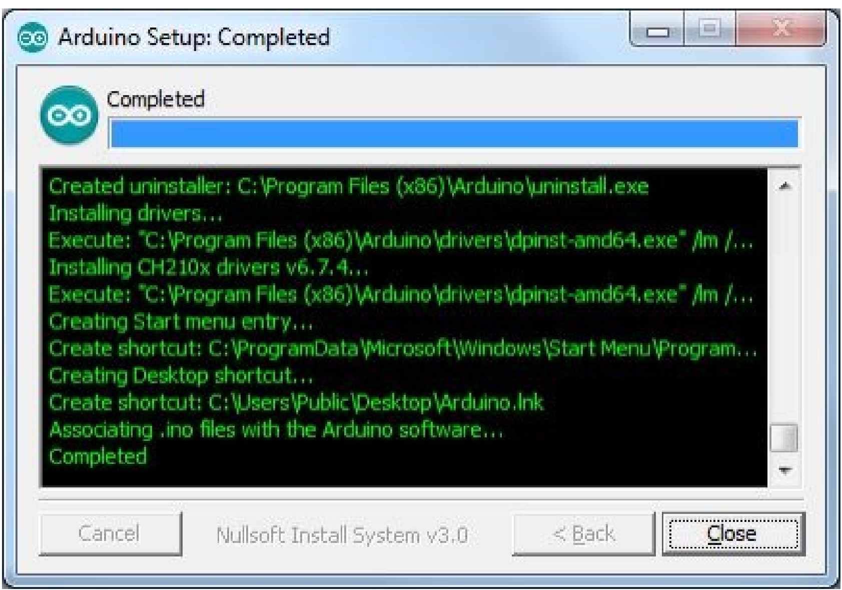


Ilustración 14. Instalando en Windows (Paso 4)

Una vez haya acabado, únicamente tienes que ejecutar la aplicación desde el acceso directo que se ha creado.

INSTALACIÓN DE ARDUINO EN LINUX

Para instalar la versión de Linux tienes que extraer el contenido del fichero

descargado en el directorio /home.

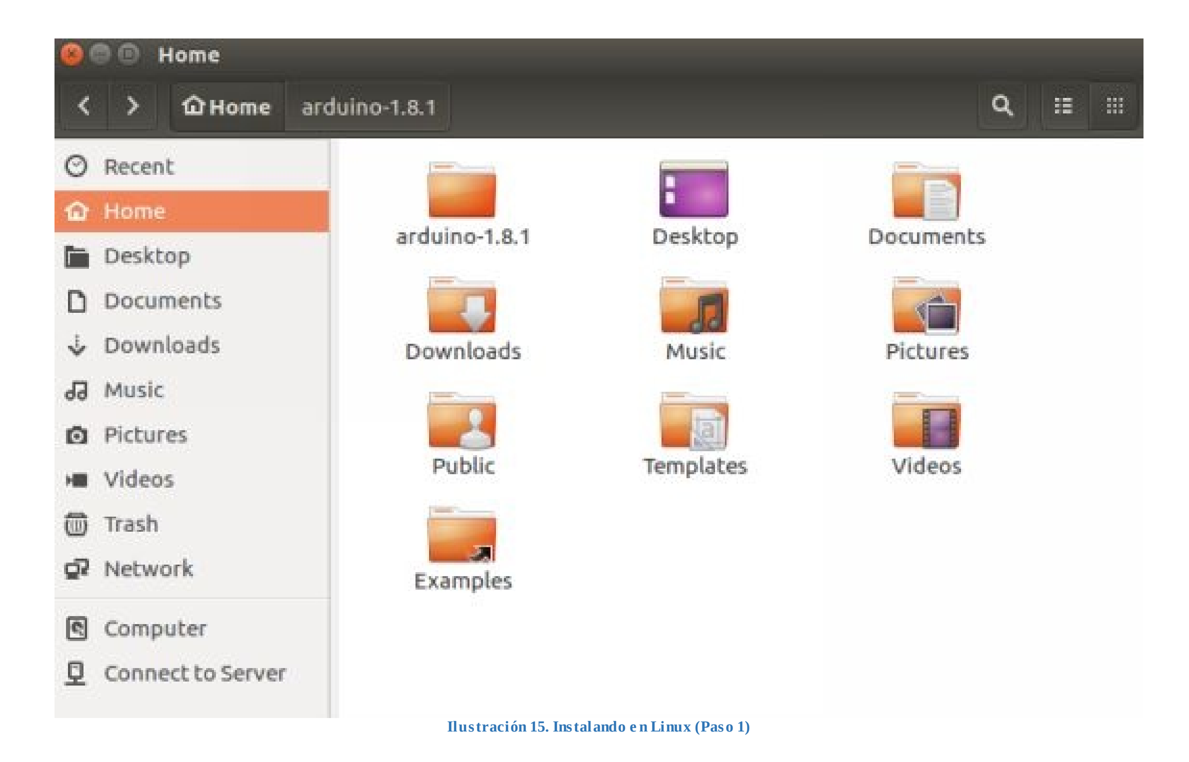


Ilustración 15. Instalando en Linux (Paso 1)

Una vez lo tienes extraído, entra para comprobar que tienes todos los ficheros y carpetas que aparecen en la siguiente imagen:

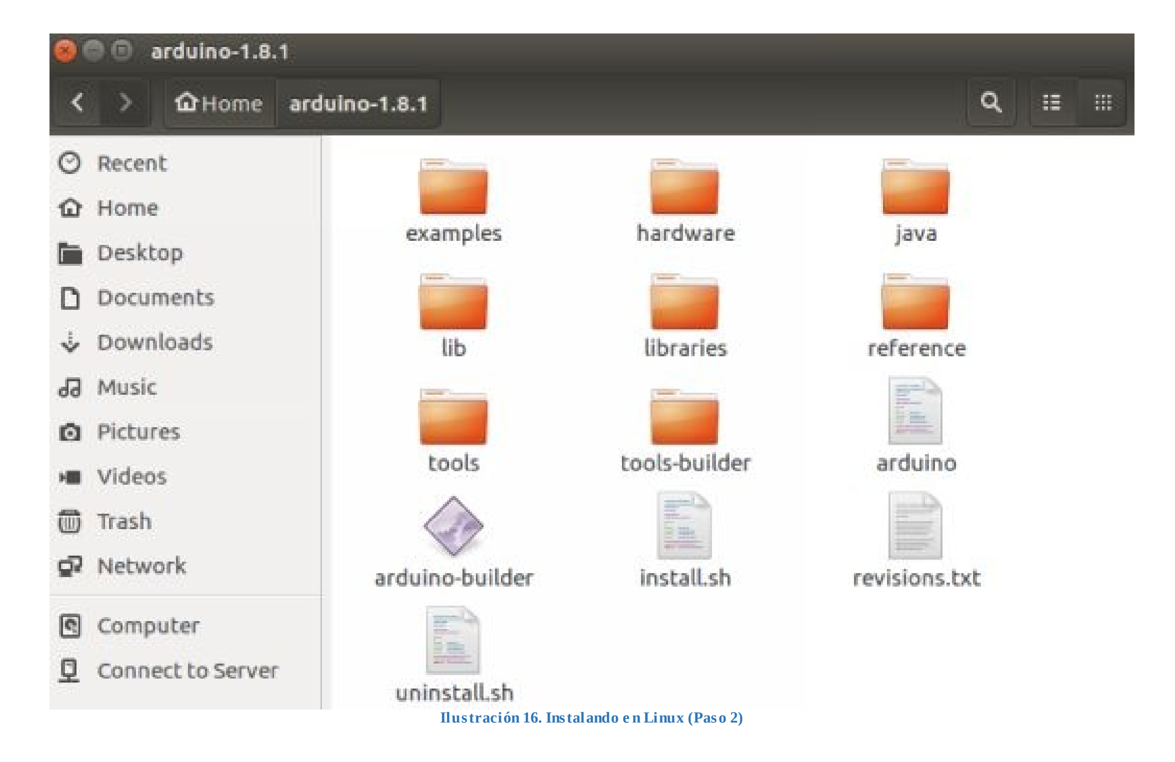


Ilustración 16. Instalando en Linux (Paso 2)

Para instalar el IDE de Arduino únicamente tienes que ejecutar el fichero *install.sh* desde el terminal:

```
timeofsoftware@timeofsoftware: ~/arduino-1.8.1
timeofsoftware@timeofsoftware:~$ ls
arduino-1.8.1  Documents  examples.desktop  Pictures  Templates
Desktop        Downloads  Music             Public    Videos
timeofsoftware@timeofsoftware:~$ cd arduino-1.8.1/
timeofsoftware@timeofsoftware:~/arduino-1.8.1$ sh install.sh
Adding desktop shortcut, menu item and file associations for Arduino IDE... done
!
timeofsoftware@timeofsoftware:~/arduino-1.8.1$
```

Ilustración 17. Instalando en Linux (Paso 3)

El proceso de instalación creará el acceso directo en el escritorio, por lo que para entrar al entorno únicamente tienes que ejecutarlo desde ahí.

En este apartado se explicará el entorno de desarrollo con el que realizarás todos los objetivos, retos y proyectos que te proponemos en este libro.

PANTALLA PRINCIPAL

Nada más abrir el entorno puedes apreciar lo sencillo e intuitivo que es. Por defecto, se te abre un proyecto vacío con la estructura del programa Arduino.

La pantalla principal está compuesta por:

- **Menú Principal** que contiene todas las funciones y herramientas de las que está compuesto el entorno.
- **Barra de acceso rápido** a las operaciones más comunes.
- **Panel inferior de mensajes** donde se muestran los mensajes de información y error enviados por el entorno al usuario que lo está utilizando.

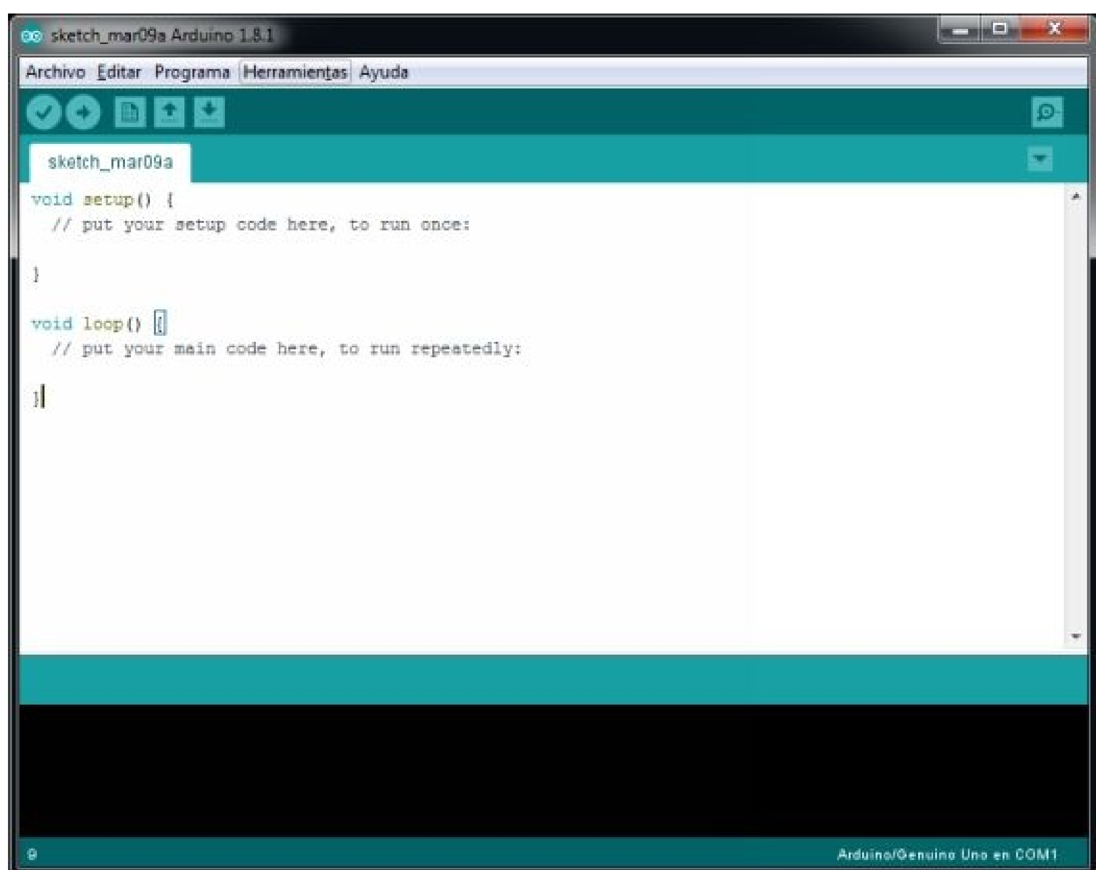


Ilustración 18. Pantalla principal IDE

MENÚ PRINCIPAL

MENÚ ARCHIVO

El menú Archivo contiene todas las funcionalidades típicas de este menú. Desde aquí se puede crear un proyecto nuevo, abrir proyectos existentes, acceder a los ejemplos predefinidos, guardar el proyecto actual.

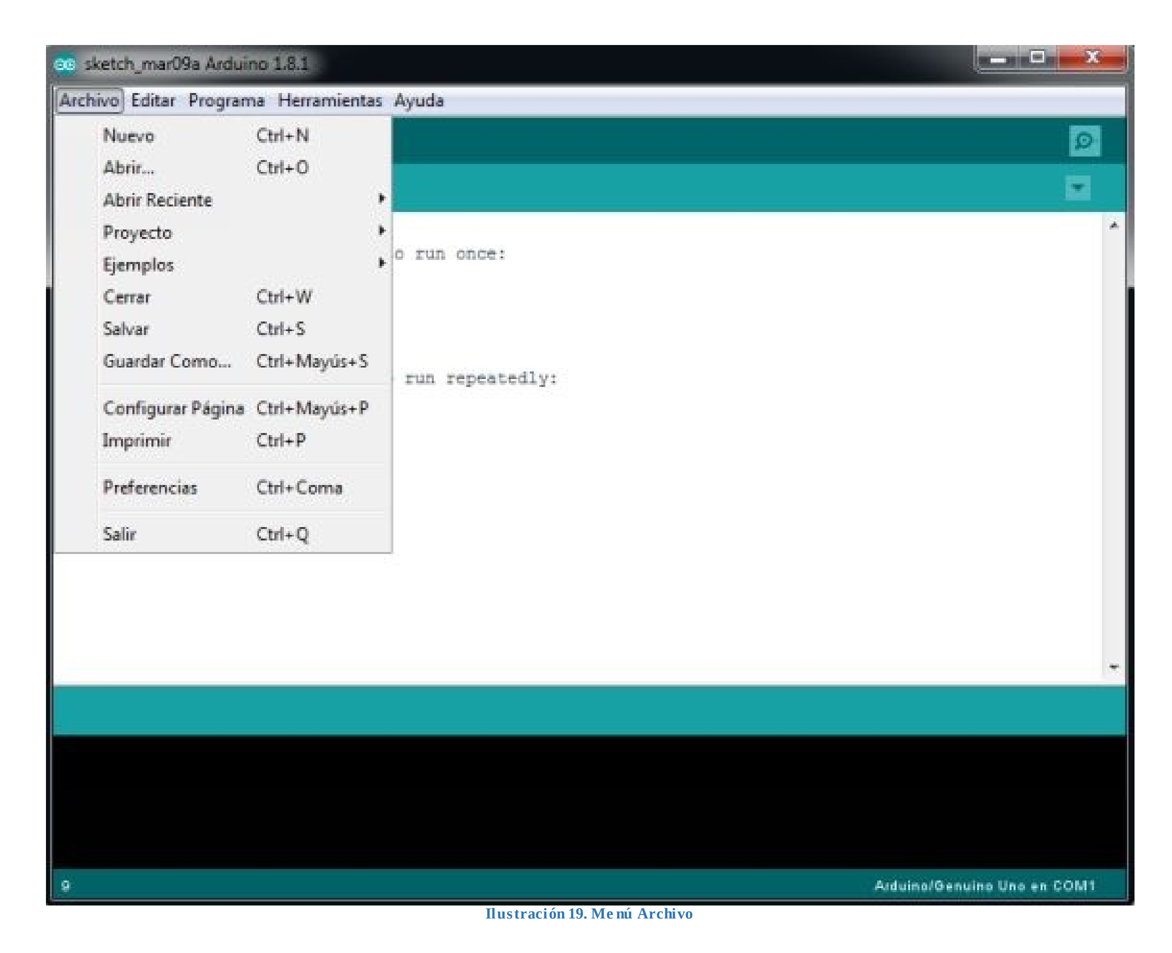


Ilustración 19. Menú Archivo

Además de las operaciones de gestión de proyectos, permite acceder al menú de configuración del entorno de desarrollo para poder configurar elementos no relacionados con el desarrollo, compilado y ejecución de los programas de Arduino.

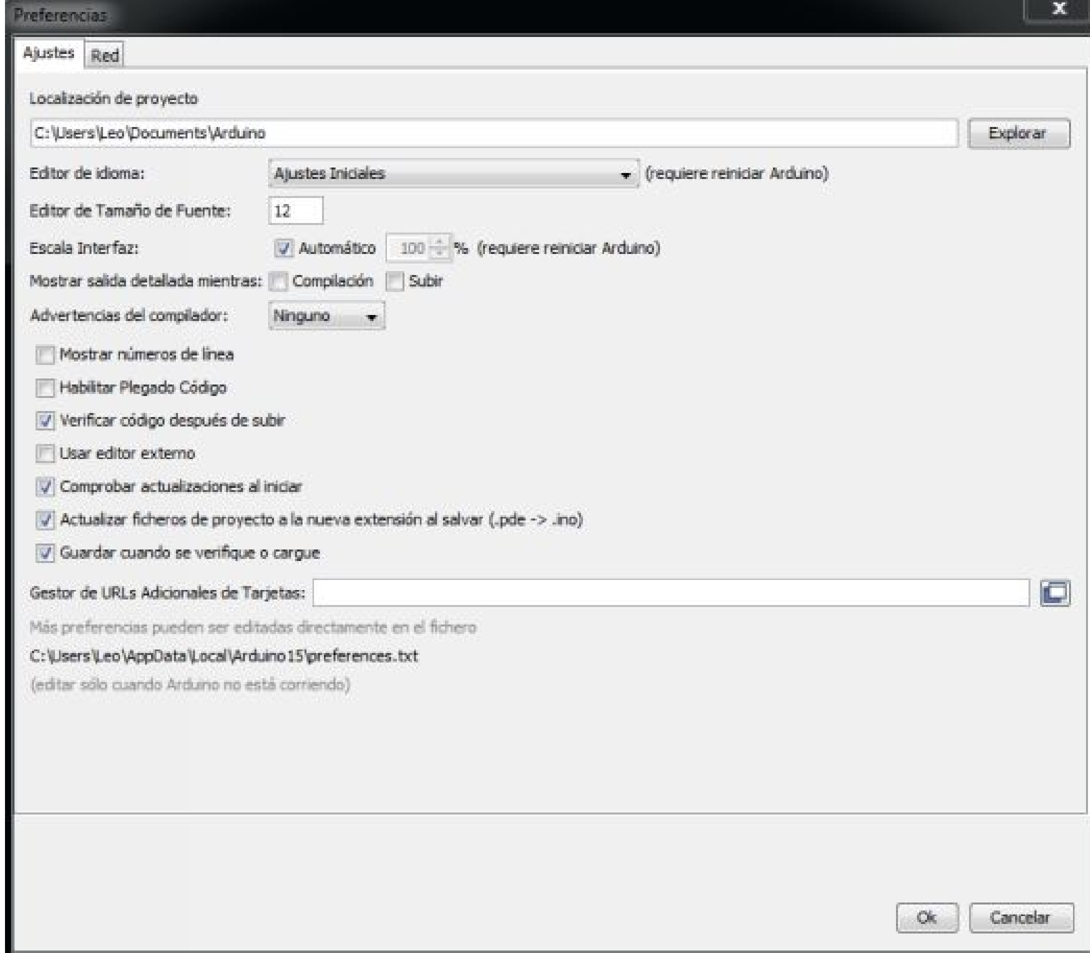


Ilustración 20. Menú preferencias IDE

MENÚ PROGRAMA

El menú Programa contiene todas las operaciones y funcionalidades que se pueden llevar a cabo con el proyecto que tenemos actualmente cargado en el entorno de desarrollo.

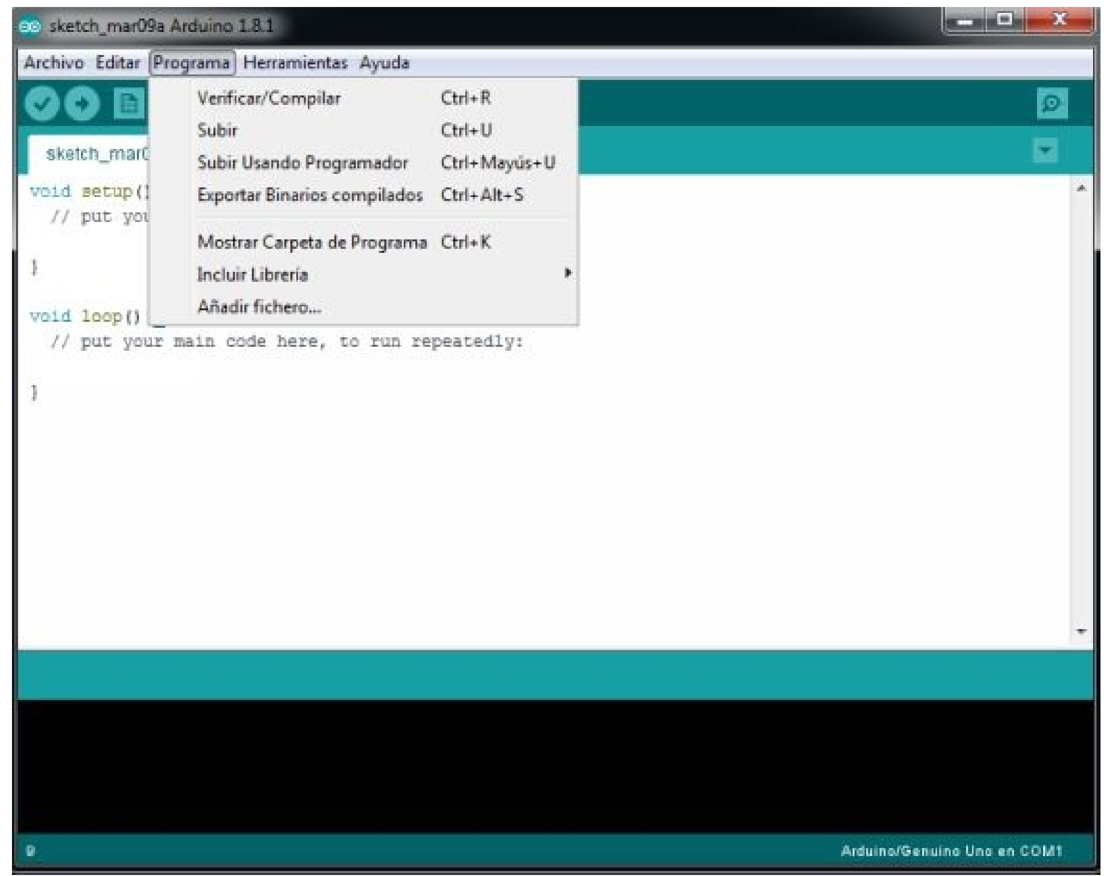


Ilustración 21. Menú Programa

Además de las funciones de compilado y carga del programa en la placa, desde este menú se pueden añadir librerías al entorno de programación. Para ello, hay que acceder a la opción e "Incluir Librería". Una vez dentro, se puede seleccionar una librería ya existente, cargar una librería en formato "Zip" o acceder a la herramienta gestora de librerías, desde donde podremos agregar librerías seleccionando la versión a instalar.

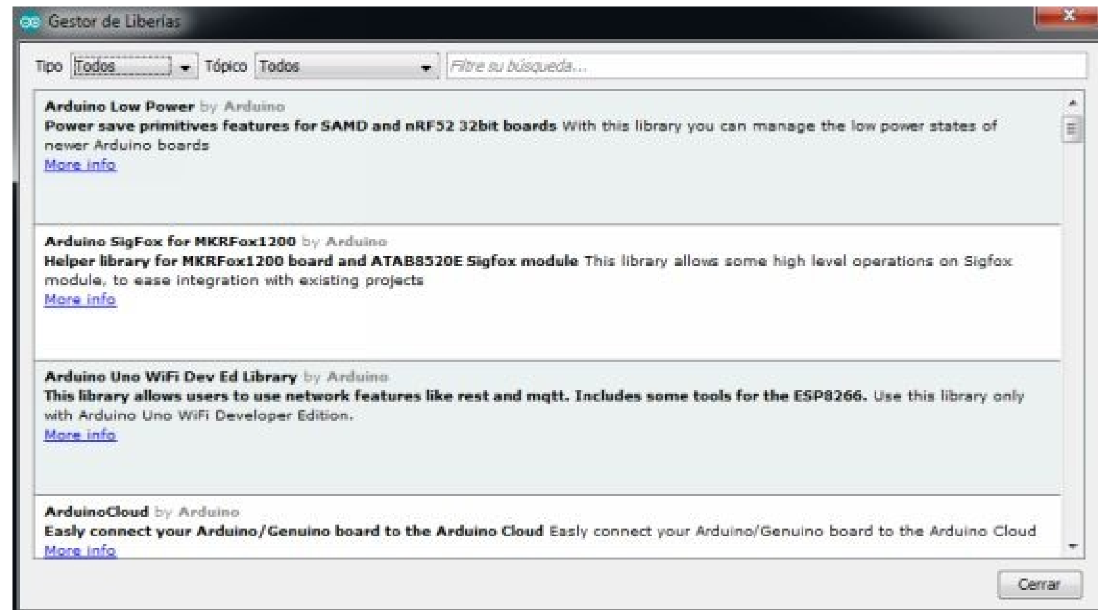


Ilustración 22. Gestor de librerías

MENÚ HERRAMIENTAS

El menú Herramientas permite el acceso y configuración de elementos ajenos al entorno de desarrollo.

Los elementos más comunes que se utilizan de este menú son:

- **Monitor Serie**: Explicado en el apartado “Pantalla Principal”.
- **Configuración de la placa**: Desde aquí se puede configurar la placa que tenemos conectada al ordenador. Por defecto la aplicación la detectará automáticamente.
- **Configuración del puerto**: Desde aquí se puede configurar el puerto de conexión de la placa al ordenador. Por defecto la aplicación la detectará automáticamente.

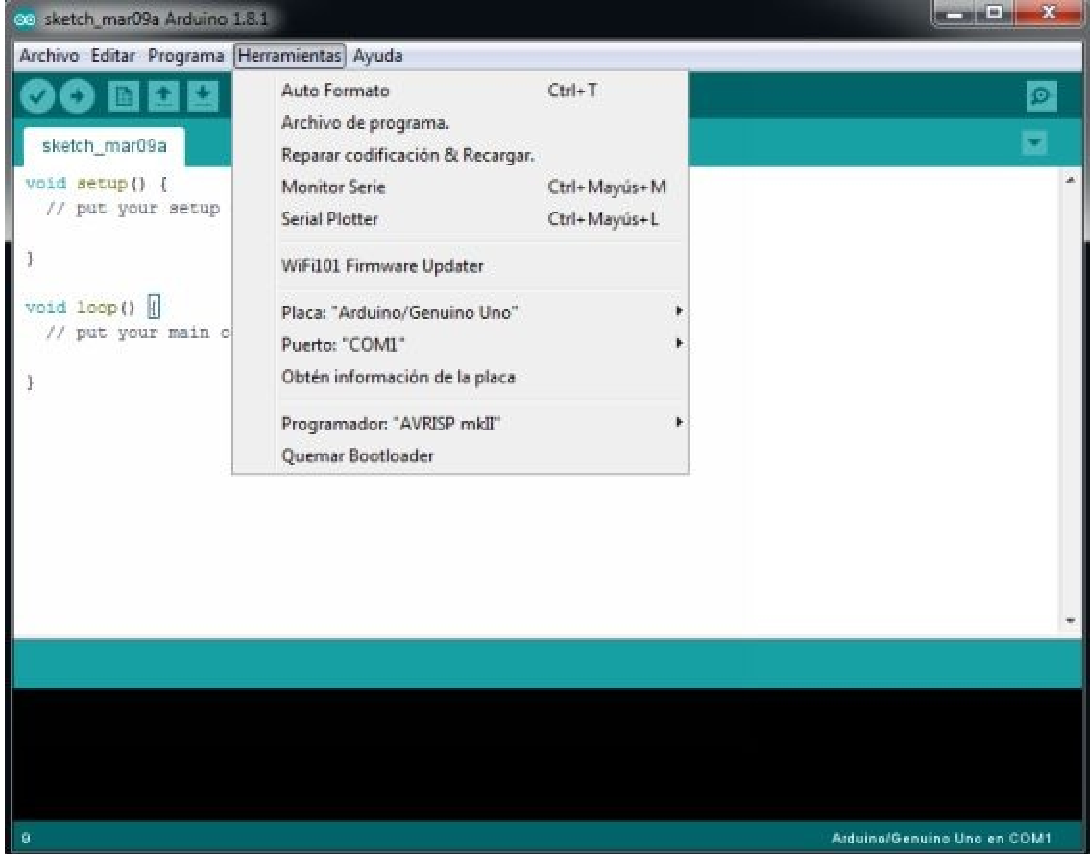
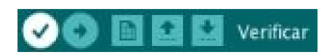


Ilustración 23. Menú Herramientas

BARRA DE ACCESO RÁPIDO

A continuación, están explicadas todas las funciones de acceso rápido incluidas en la barra de accesos directos.

VERIFICAR



La operación *"Verificar"* permite comprobar si el código fuente escrito en el entorno de desarrollo no tiene fallos, es decir, comprueba si compila.

El resultado de la verificación del código es mostrado en la consola de mensajes inferior, si no contiene errores la consola mostrará un mensaje como este:

```
Compilado

El Sketch usa 444 bytes (1%) del espacio de almacenamiento de programa. El máximo es 32256 bytes.
Las variables Globales usan 9 bytes (0%) de la memoria dinámica, dejando 2039 bytes para las variables locales. El máximo es 2048 bytes.
```

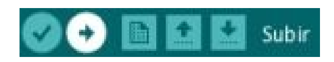
Ilustración 24. Verificar OK

En caso de contener errores, todos ellos serán mostrados de la siguiente forma:

```
base operand of '->' has non-pointer type 'MFRC522::Uid'
/Users/alfre/Desktop/RFID/RFID.ino: In function 'void loop()':
RFID:84: error: base operand of '->' has non-pointer type 'MFRC522::Uid'
   for (byte i = 0; i < &(mfrc522.uid)->size; i++) {
                                      ^
RFID:85: error: base operand of '->' has non-pointer type 'MFRC522::Uid'
     if(&(mfrc522.uid)->uidByte[i] < 0x10)
                      ^
RFID:89: error: base operand of '->' has non-pointer type 'MFRC522::Uid'
     Serial.print(&(mfrc522.uid)->uidByte[i], HEX);
                                ^
RFID:95: error: base operand of '->' has non-pointer type 'MFRC522::Uid'
   if(&(mfrc522.uid)->sak < 0x10)
                    ^
RFID:100: error: 'PICC_Type' was not declared in this scope
   PICC_Type piccType = PICC_GetType(&(mfrc522.uid)->sak);
   ^
RFID:102: error: 'piccType' was not declared in this scope
```

Ilustración 25. Verificar KO

SUBIR



La operación *"Subir"* permite cargar el código fuente compilado en la placa de Arduino. Una vez subido el programa, éste comenzará a ejecutarse en la placa.

NUEVO



La operación *"Nuevo"* abrirá un nuevo proyecto de Arduino vacío, con la estructura principal ya creada.

ABRIR



La operación *"Abrir"* permite buscar un proyecto Arduino en nuestro ordenador y abrirlo en el entorno de desarrollo.

SALVAR



La operación *"Salvar"* guardará el proyecto de Arduino en la ruta donde le indiquemos.

MONITOR SERIE



La operación *"Monitor Serie"* abrirá la consola de intercambio de mensajes entre la placa y el ordenador a través del puerto de comunicaciones.

Tal y como podrás ver en una fase del objetivo número 4, el "Monitor *Serie"* se utiliza para mostrar información en la pantalla del ordenador.

ESTRUCTURA DE UN PROGRAMA

El programa en Arduino es muy sencillo. Está compuesto básicamente por dos secciones diferentes y obligatorias en las que se codifican diferentes componentes del programa.

```
void setup() {

}

void loop() {

}
```

Ilustración 26. Estructura programa Arduino

En la primera sección del programa, **setup()**, es donde se define la configuración del programa, es decir, es donde se codifican todas las sentencias de configuración, inicialización de los puertos de la placa, configuración de las comunicaciones y otros parámetros que dependerán de cada programa que se desarrolle.

En la segunda sección del programa, **loop()**, es donde se codifica el programa que se ejecutará de forma cíclica en Arduino. Esta sección está compuesta por todas las sentencias que componen el programa y que llevarán a cabo todas las operaciones que lo compongan.

Resumiendo, en **setup** se define e inicializa lo que se usa en **loop**, siendo esta última sección el corazón de un programa de Arduino.

COMPONENTES COMUNES EN TODOS LOS MONTAJES

PLACA ARDUINO

La placa Arduino ha sido explicada en el punto anterior en el que te hemos explicado Arduino. En los ejercicios que se realizarán posteriormente, se irán definiendo las conexiones en función del montaje que estemos llevando a cabo.

PROTOBOARD

La placa *protoboard* es una base de material aislante (plástico) con orificios para la inserción de los componentes electrónicos, de forma que estén unidos eléctricamente entre ellos, ya que en su interior los orificios están conectados entre sí gracias a unas líneas de material conductor.

Usar la placa Protoboard es muy sencillo. Simplemente tienes que insertar los componentes teniendo en cuenta que hay líneas que actúan como si todos los componenentes estuvieran conectados en el mismo punto.

Las dos líneas de orificios externas, que se encuentran entre la línea pintada roja y azúl, destacada una de ellas en amarillo, actúan como si fuera el mismo punto de conexión, es decir, cualquier componente que pinchemos en cualquiera de los orificios señalados sería como si estuviera conectado directamente a otro componente que estuviera pinchado en otro de los puntos de la misma línea.

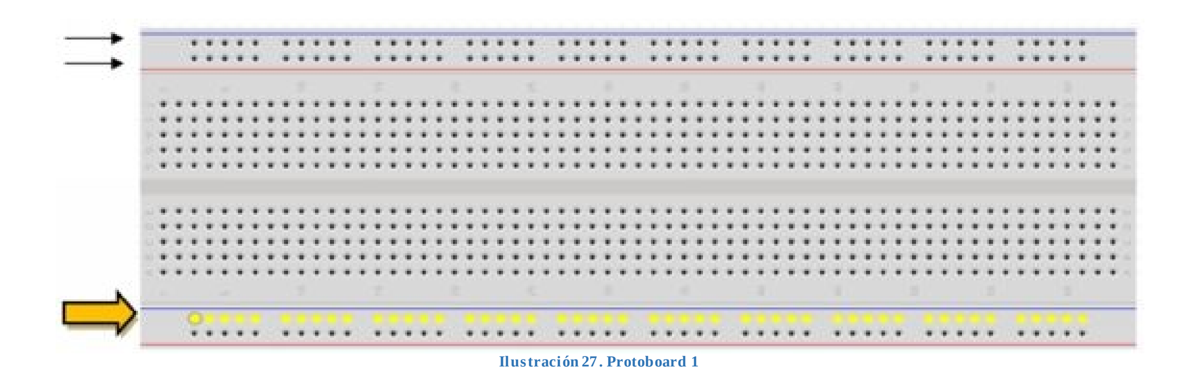


Ilustración 27. Protoboard 1

La línea azul y la línea roja nos pueden ayudar para conectar en uno de sus extremos un cable con carga negativa y positiva respectivamente y tener así el resto de los orificios alimentados con dichas cargas.

Los orificios de la parte central de la placa están conectados eléctricamente de forma diferente a las líneas exteriores. En la imagen siguiente puedes ver una línea iluminada de amarillo que son los puntos conectados eléctricamente entre sí.

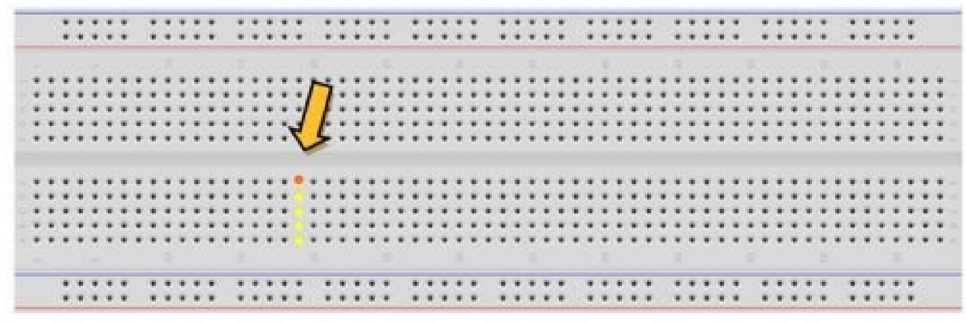


Ilustración 28. Protoboard 2

Ojo, para conectar una mitad de la placa con la otra, tendremos que hacer un puente de forma que conecte a través de un cable una línea con la otra como vemos en la imagen siguiente:

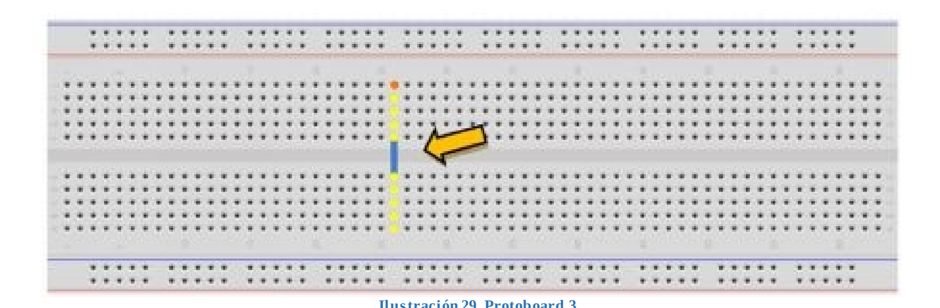


Ilustración 29. Protoboard 3

CABLE USB

Lo usaremos para pasar el programa desde el ordenador a la placa de Arduino, para que interactúe con los componentes electrónicos, y en sentido inverso, desde los componentes electrónicos (sensores), al ordenador.



Ilustración 30. Cable USB

CABLES

Usaremos cables macho-macho o macho-hembra en función de lo que necesitemos conectar en cada proyecto.



Ilustración 31. Cables de circuito

RESISTENCIAS

Las resistencias son unos dispositivos que se oponen al paso de la corriente eléctrica cuando pasa través de ellas, por tanto, antes de una resistencia y después de ella habrá una diferencia de carga eléctrica, siempre menor a la salida que a la entrada. De esta manera, protegeremos a otros dispositivos de ser atravesados por una intensidad mayor a la pueden soportar y por tanto sufrir una sobrecarga.

Para saber qué resistencia será necesaria para proteger un determinado dispositivo, necesitaremos conocer la intensidad máxima que puede soportar el dispositivo (I) y la tensión a la que va a estar sometido (V). Conociendo dichos valores podemos calcular la resistencia (R) que protegerá nuestro dispositivo. Para calcularlo utilizamos la LEY DE OHM (**V=I x R):**

$$R = V / I$$

Donde la resistencia (R) se mide en ohmios, **Ω**; la tensión eléctrica (V) se mide en voltios, **V**; y la corriente o intensidad eléctrica (I) se mide en amperios, **A**.

Algo que debes tener en cuenta es que **no existen resistencias comerciales de todos los valores óhmicos**, por lo que tendrás que calcular y elegir la resistencia que más se aproxime al valor ideal del proyecto que estás

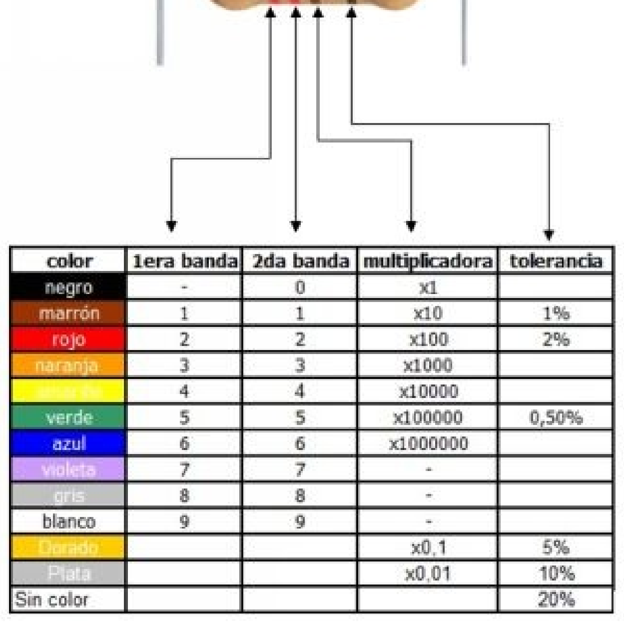
realizando. Por ello, a continuación, te explico un ejemplo de cómo calcular el valor de una resistencia para proteger un dispositivo, en este caso, un LED.

Tenemos que tener en cuenta dos cuestiones: la primera es que un LED estándar soporta una intensidad de hasta **20mA**, y dejaremos un margen para no correr el riesgo de que el LED se funda (17mA). Y la segunda es que el voltaje con el que vamos a trabajar es de unos **5V**. Con ello, vamos a calcular el valor de la resistencia que tendríamos que montar en serie para que el LED no sufra una sobrecarga.

Aplicando la ley de Ohm:
V = I x R
5V = 17mA x R
R = 5/0,017
R = 294 Ω

Puesto que no vamos a encontrar una resistencia con este valor exacto, tendrás que elegir otra de valor aproximado siempre por encima, para asegurarnos la protección del dispositivo. En este caso nos tendremos que hacer con una resistencia de 330 Ω.

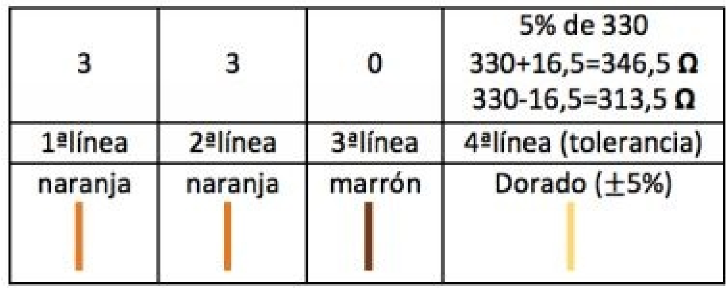
Una vez que hemos calculado el valor de la resistencia que necesitamos, debemos elegir físicamente esa resistencia. Pero, ¿cómo podemos saber cuál es el valor óhmico de una resistencia física? Muy sencillo: fijándonos en su código de colores. Cada resistencia cuenta con unas líneas de colores que según su combinación ofrecen más o menos resistencia.



color	1era banda	2da banda	multiplicadora	tolerancia
negro	-	0	x1	
marrón	1	1	x10	1%
rojo	2	2	x100	2%
naranja	3	3	x1000	
amarillo	4	4	x10000	
verde	5	5	x100000	0,50%
azul	6	6	x1000000	
violeta	7	7	-	
gris	8	8	-	
blanco	9	9	-	
Dorado			x0,1	5%
Plata			x0,01	10%
Sin color				20%

Ilustración 32. Relación bandas resistencia y color

En este caso, la resistencia que necesitábamos, de 330 **Ω**, sería la que tiene los colores:



3	3	0	5% de 330 330+16,5=346,5 Ω 330-16,5=313,5 Ω
1ªlínea	2ªlínea	3ªlínea	4ªlínea (tolerancia)
naranja	naranja	marrón	Dorado (±5%)

Ilustración 33. Colores resistencia utilizada

LED

L.E.D. (Light Emitting Diode o diodo emisor de luz), es un dispositivo que emite luz cuando se encuentran en polarización directa, es decir necesita estar conectado de una forma determinada: la pata larga (**ánodo**) conectada a un polo positivo y la pata corta (**cátodo**) a un polo negativo. Si se conecta al revés, la corriente no pasará y por tanto no emitirá luz. Esto es algo muy importante que debes tener en cuenta a la hora de realizar los montajes de tus proyectos.

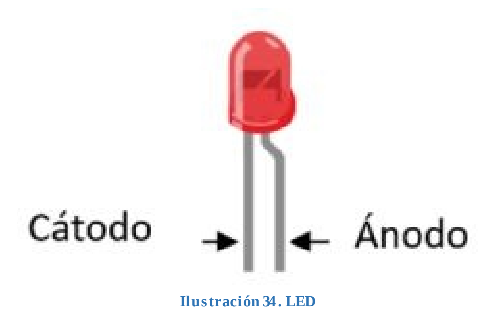


Ilustración 34. LED

Los LEDs se utilizan mucho para realizar montajes electrónicos, ya que presentan dos ventajas importantes con respecto a otro tipo de dispositivo lumínico:

1. Consumen muy poca energía ya que casi toda la energía que consume un LED (más de un 80%) se utiliza en dar luz y no calor.

2. Tienen una vida útil muy larga: más de 100.000 horas de luz.

Un LED puede conectarse tanto a un pin digital como analógico.

LED RGB

Un LED especial es el RGB, cuyas siglas hacen referencia a los colores Red (Rojo), Green (Verde) y Blue (Azul), los colores primarios de la luz. Un RGB es en realidad la unión de tres LEDs, cada uno de un color básico, en un encapsulado común. En función de la tensión que reciba cada uno de ellos, el RGB emitirá un color u otro.

Podemos encontrar dos tipos de RGB:

- De cátodo común: comparten el pin que se conecta a GND. Los otros tres pines se conectan a pines digitales o analógicos.

- De ánodo común: comparten el pin que se conecta a la alimentación (5V en nuestra placa de Arduino). Los otros tres pines se conectan a pines

digitales o analógicos.

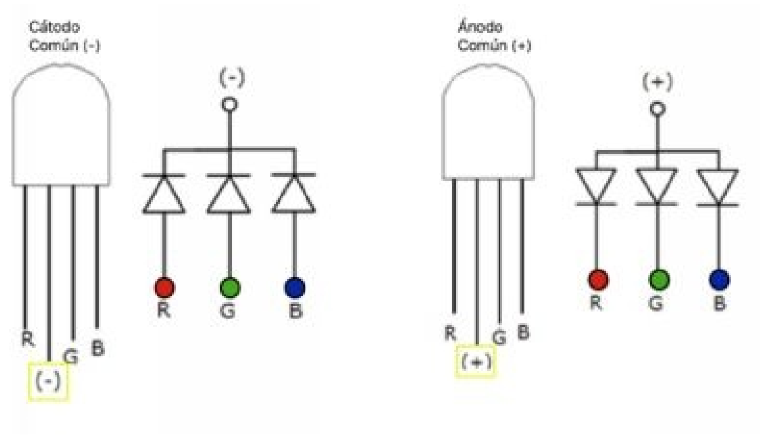


Ilustración 35. Tipos de LED RGB

Para identificar qué pin corresponde a cada color debemos tener en consideración:

- La patilla más larga es la de alimentación o la tierra (GND), según sea cátodo común o ánodo común.

- Normalmente, la patilla que está sola al lado de la más larga es el rojo R y las otras dos corresponden al verde G y al azul B.

Teniendo en cuenta que el rango de valores RGB va de 0 a 255, podemos jugar a combinar unos colores con otros.

La elección de dichos valores se traduce como la tensión que podemos poner en cada pin. Mediante la elección de los valores conseguiremos la mezcla de color que deseemos de forma sencilla.

Con la siguiente imagen puedes hacerte una idea de la luz que obtendremos si mezclamos varias de ellas, te animamos a que pruebes a introducir diferentes valores para cada uno de los colores y así podrás comprobar diferentes tonalidades de colores.



Ilustración 36. Luz

Tal y como puedes observar, la mezcla de la luz de color rojo, verde y azul nos da como resultado una luz blanca.

POTENCIÓMETRO

Un potenciómetro es un dispositivo que proporciona una resistencia variable según vayamos modificando su posición gracias a la rotación de un eje que se va desplazando interiormente sobre un material resistivo. Un ejemplo de potenciómetro es el que vemos en la siguiente imagen, aunque existen en el mercado muchos tipos más.

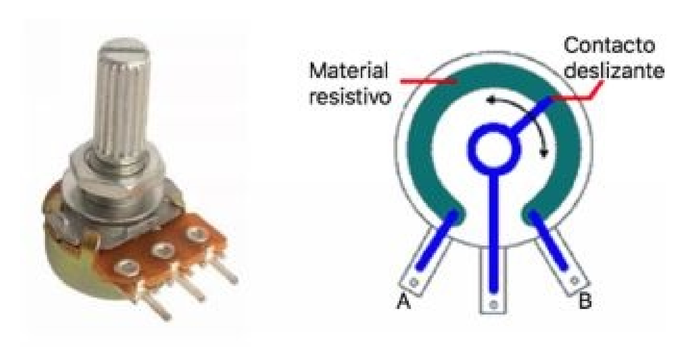


Ilustración 37. Potenciómetro

Para conectar el potenciómetro a la placa Arduino, debemos unir la patilla A a GND (tierra), la B a los 5V (o viceversa) y la central a una entrada analógica, ya que lo valores que vamos a introducir van a ser variables.

ZUMBADOR

Un zumbador es un dispositivo fabricado con un material piezoeléctrico, normalmente cuarzo, que al ser sometido a una tensión eléctrica variable vibran produciendo un sonido.



Ilustración 38. Zumbador

La conexión del zumbador es sencilla, una pata conectará con GND (tierra) y la otra a una entrada analógica o digital por donde entrará la tensión que le provocará la vibración.

SENSOR DE LUZ (LDR)

El LDR (Light Dependent Resistor) es una resistencia cuyo valor depende de la cantidad de luz que incida sobre ella.



Ilustración 39. Sensor de luz

Cuando la luz incide sobre el LDR, el material se vuelve más conductor por lo que su resistencia disminuye. Al contrario, cuanta menos luz incida sobre ella, mayor será su resistencia. Ésto se traduce en que podrá dar paso a mayor o menor cantidad de corriente eléctrica en función de la luz que reciba.

Al ser un dispositivo que no tiene polaridad, las patillas pueden conectarse indistintamente, sin tener en cuenta el polo positivo ni el negativo.

SENSOR DE HUMEDAD Y TEMPERATURA

El DHT11, *Digital Temperatura and Humidity sensor*, es un dispositivo que nos va a permitir obtener una información simultánea de temperatura y humedad mediante un procesador interno que realiza dicho proceso de medición.

El sensor que vamos a utilizar es el DHT11, no es el mejor de todos, pero para realizar nuestro proyecto es más que suficiente. Si queremos afinar más en nuestras mediciones podríamos hacernos con un DHT22, más preciso, pero también más caro.

Las características técnicas del DHT11 son las siguientes, teniendo en cuenta que el sensor va tomando valores ambientales cada segundo:

- Medición de Tª de 0 a 50°C ± 2°C
- Medición de H de 20 a 80% ± 5%

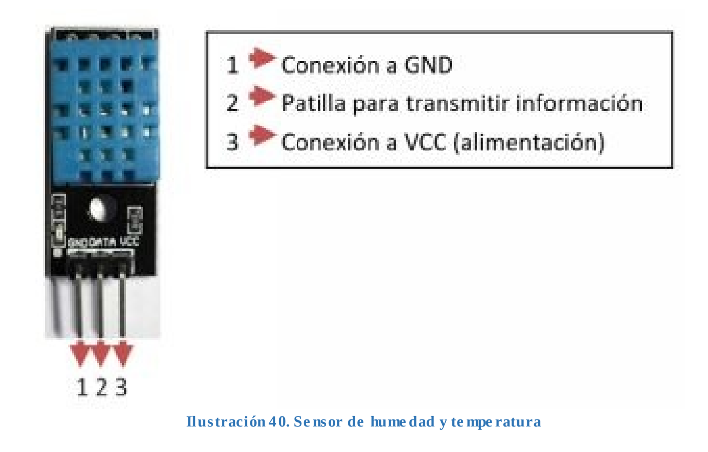


Ilustración 40. Sensor de humedad y temperatura

Para conectar el sensor a la placa Arduino lo tendremos fácil. Fijándonos en la leyenda del dispositivo, conectamos las patillas exteriores a GND y VCC y la central a una entrada digital, ya que el procesador del DHT11 proporciona mediciones mediante señales digitales.

SENSOR DE PRESENCIA

El sensor infrarrojo de movimiento PIR HC-SR501, donde PIR viene de ‘Pyroelectric Infrared’ o ‘Passive Infrared’, es un dispositivo que detecta movimiento basándose en la radiación electromagnética infrarroja que emiten los seres vivos y objetos debido a la temperatura a la que se encuentran.

El sensor detecta presencia si el objeto se encuentra dentro de un cono de 110° desde su centro y una distancia máxima de hasta 7 metros.

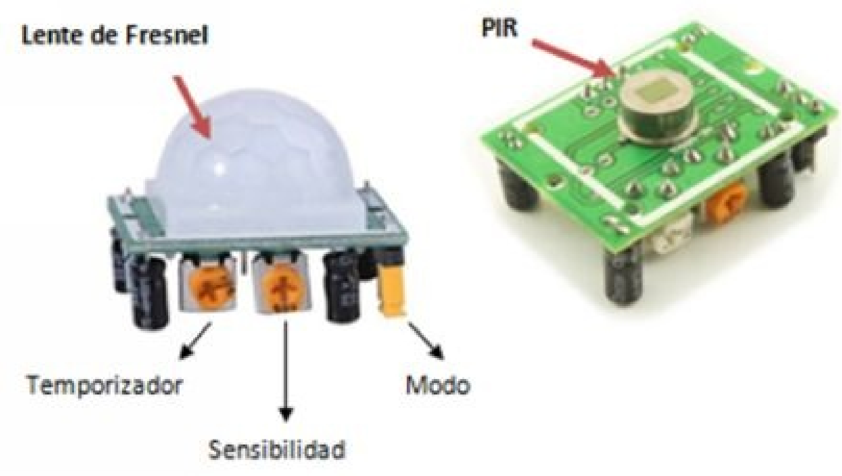


Ilustración 41. Sensor de presencia

Podemos ajustar el comportamiento del sensor modificando con un destornillador los dos potenciómetros y el jumper que incorpora:

- **Ajuste del temporizador**: Para marcar el tiempo en el que se activará el sensor cuando haya detectado presencia, de 3 segundos a 5 minutos. Girando hacia la derecha tardará más en actuar y girando hacia la izquierda se activará en menos tiempo.

- **Ajuste de sensibilidad:** Para modificar la distancia y la cantidad de movimiento necesaria para activar el sensor, de 3 a 7 metros aproximadamente. Girando hacia la derecha decrece la sensibilidad (se activará cuando detecte el objeto a menos distancia). Girando hacia la izquierda aumenta la sensibilidad (se activará cuando detecte el objeto a mayor distancia).

- **Ajuste del modo:** El sensor puede trabajar en modo continuo: si queremos esté continuamente detectando movimiento y por tanto activado, o en modo repetición: si queremos que sólo se active cuando detecte movimiento.

Para conectarlo con la placa de Arduino: conectamos el pin VCC del sensor a los 5V de la placa, el pin central del sensor a un pin de la placa para transmitir información y el pin GND del sensor al GND de Arduino.

SENSOR DE AGUA

Lo primero de todo es no confundir el sensor de agua con el sensor de humedad, ya que miden cosas diferentes.



Ilustración 42. Sensor de agua

Este sensor estima la presencia de agua a través de unos filamentos conductores paralelos que serán los que entren en contacto directo con el líquido. Esto provoca una señal analógica que será traducida por Arduino indicando que efectivamente el sensor ha detectado agua. Dichos valores analógicos medidos pueden variar desde 0 (sensor totalmente seco) a 1023 (sensor totalmente empapado).

Aunque el sensor necesita entrar en contacto con el agua para detectar su existencia o no, eso no quiere decir que todas sus partes puedan mojarse. ¡No olvidemos que es un dispositivo electrónico! Por ello debemos proteger los contactos y partes eléctricas del circuito. De lo contrario, podremos provocar un cortocircuito.

PANTALLA LCD

La pantalla LCD, Liquid Crystal Display (Display de cristal líquido), es un dispositivo que nos va a permitir visualizar información de forma gráfica mediante texto.

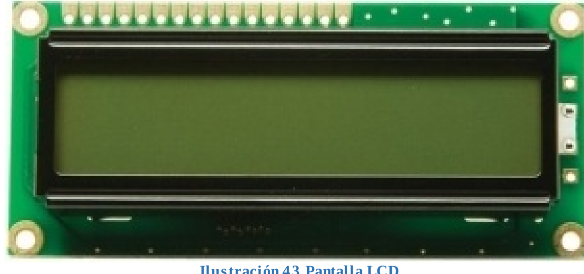


Ilustración 43. Pantalla LCD

Estas pantallas poseen pines de entrada/salida de datos como se muestra en la siguiente imagen.

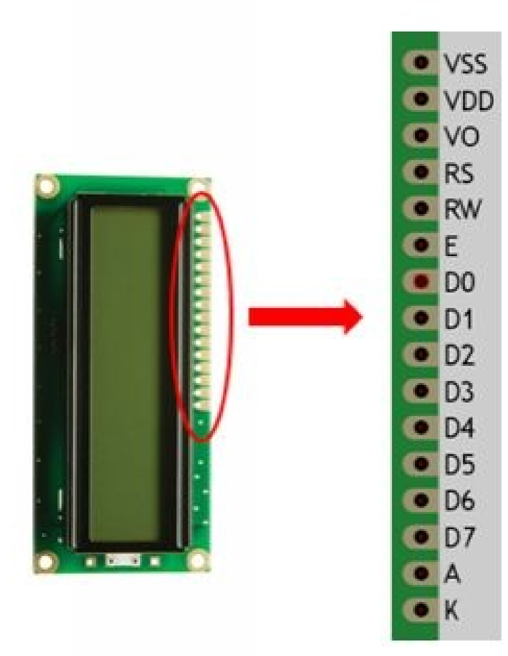


Ilustración 44. Detalle pines LCD

El significado de cada uno de los pines de la pantalla LCD es el siguiente:

- **VSS**: pin que conecta a tierra (GND, masa)
- **VDD**: pin que conecta a 5 voltios para alimentar la pantalla.
- **VO**: pin que ajusta el contraste de la pantalla. Se conecta a un potenciómetro normalmente.
- **RS**: pin selector de registro para elegir el uso del dispositivo.
- **RW**: pin de lectura/escritura. En función del estado (HIGH o LOW), podremos escribir o leer datos en el LCD.
- **E** (enable): pin que habilita/deshabilita la pantalla para recibir información.
- **D0** a **D7**: pines de datos por donde se envía o recibe información, es decir, por donde se transfieren los datos.
- **A**: pin del LED (ánodo) que ilumina la pantalla (5V)
- **K**: pin del LED (cátodo) que ilumina la pantalla (GND)

De forma gráfica, podemos ver la conexión de la placa Arduino con la pantalla LCD en la siguiente imagen. A la izquierda se encuentra la placa Arduino y a la derecha la pantalla LCD. Las líneas de colores indican las

uniones que se deben hacer para conectar ambos dispositivos:

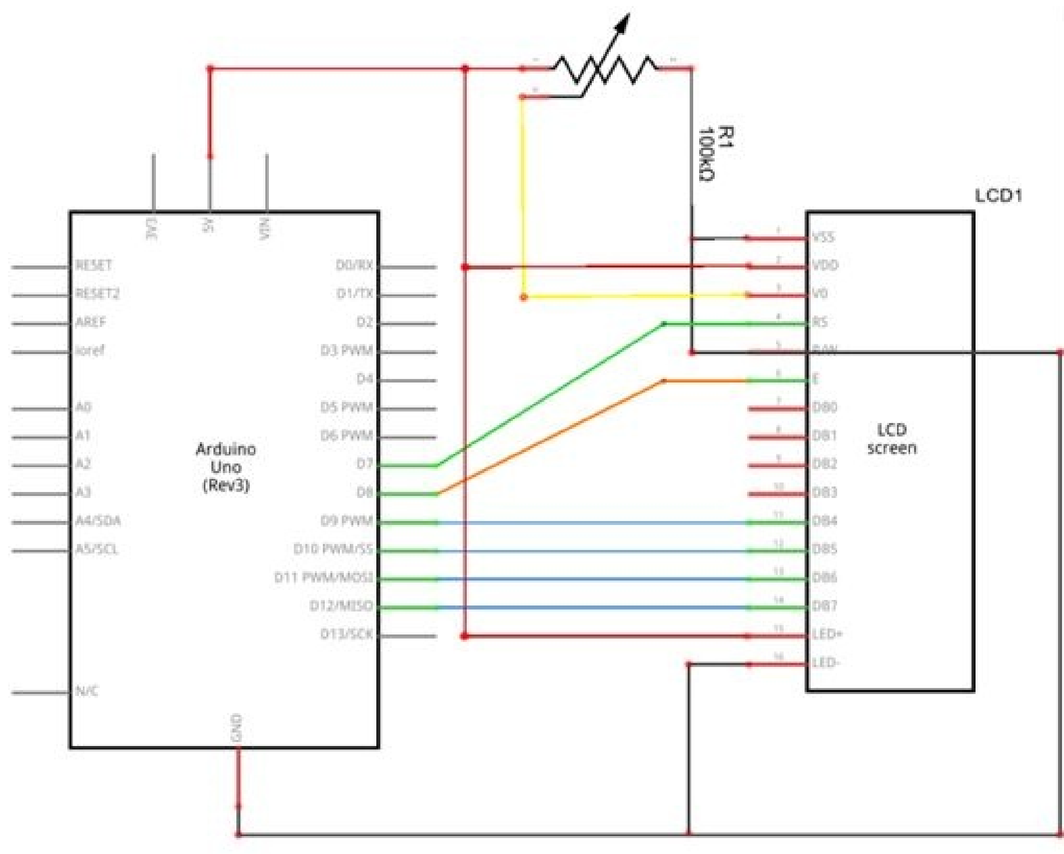


Ilustración 45. Conexión entre placa Arduino y LCD

En el Objetivo 4 utilizaremos la pantalla LCD para visualizar la información que nos aportan determinados sensores y las conexiones vendrán descritas para cada uno de ellos.

EXPLICACIÓN

En el objetivo número 1 explicaremos todo lo necesario para aprender a manejar diferentes tipos de LEDs, interactuando con diferentes pines de salida de la placa de Arduino.

FUNCIONES DE PROGRAMACIÓN

En este objetivo vamos a profundizar en las siguientes funciones de programación:

#DEFINE

Esta sentencia permite la definición de valores constantes en todo el programa. Por ejemplo, si en el circuito vamos a utilizar la salida número 5 de la placa, lo normal es crear una constante con el valor 5 y utilizarla en el código en lugar de utilizar el valor 5.

La utilización de constantes en el código es algo muy común que se utiliza en programación. En el desarrollo de programas para Arduino nos va a permitir, entre otras cosas, que, si por ejemplo cambiamos el número del pin en la que tenemos enchufado el circuito, cambiando el valor de la constante, de forma automática en todas las sentencias en las que estamos usando ese valor se actualicen y todo siga funcionando de forma correcta. Es decir, no tendremos que cambiar por el nuevo valor todas las sentencias donde se utiliza la salida, únicamente cambiaremos el valor de la constante.

En el siguiente ejemplo se define una constante que se llama VERDE y que tiene el valor 3, es decir, el LED verde está conectado al pin número 3:

#define VERDE 3

PINMODE

La sentencia es utilizada para configurar el modo de trabajo de cada uno de los pines que tiene la placa Arduino. La sección en la que se utiliza es la sección de configuración, es decir, en *setup()*.

Existen dos modos diferentes de funcionamiento de los pines:

- **INPUT**: El pin es utilizado como entrada de información en la placa de Arduino.

- **OUTPUT**: El pin es utilizado como salida de información de la placa de Arduino.

La configuración por defecto de todos los pines de la placa de Arduino es de salida, por tanto, si no indicamos lo contrario estos pines funcionarán en este modo. Te aconsejamos que, independientemente de que sea el valor por defecto el de salida, configures el modo de funcionamiento de todos los pines.

En el siguiente ejemplo se define el pin número 3 como salida:

pinMode(3, OUTPUT);

En lugar de utilizar el valor 3, en el siguiente ejemplo te mostramos como utilizar la constante que hemos definido previamente:

pinMode(VERDE, OUTPUT);

DIGITALWRITE

Mediante esta sentencia vamos a enviar valores binarios (0 o 1) al pin configurado como salida. En la sentencia hay que indicar el número del pin, que va de 0 a 13 y puede especificarse por valor o por constante, y el valor a enviar. Los posibles valores a enviar son:

- HIGH: Para enviar el valor 1.
- LOW: Para enviar el valor 0.

En el siguiente ejemplo se envía el valor HIGH al pin número 3, es decir, quiero activar el dispositivo conectado al pin 3, en este caso, encender el led conectado al pin 3:

digitalWrite(3, HIGH);

En lugar de utilizar el valor 3, en el siguiente ejemplo te mostramos como utilizar la constante que hemos definido previamente:

digitalWrite(VERDE, HIGH);

ANALOGWRITE

La sentencia enviará valores entre 0 y 255 al pin configurado como salida. En la sentencia hay que indicar el pin sobre el que se quiere realizar la escritura y el valor a escribir.

En el siguiente ejemplo se envía el valor 120 al pin número 3:

```
analogWrite(3, 120);
```

En lugar de utilizar el valor 3, en el siguiente ejemplo te mostramos como utilizar la sentencia con la constante que hemos definido previamente:

```
analogWrite(VERDE, 120);
```

DELAY

La función permite detener de forma temporal la ejecución del programa de Arduino. El tiempo de detención se indica en milisegundos.

En el siguiente ejemplo se detiene el programa medio segundo (ten en cuenta que un segundo son 1000 milisegundos):

```
delay(500);
```

FOR

La sentencia FOR es utilizada para repetir un bloque de sentencias un número determinado de veces, es decir, utilizaremos esta sentencia si queremos repetir una acción o secuencia de acciones, como por ejemplo que se encienda y se apague un led varias veces seguidas.

Para la ejecución se utiliza una condición de terminación, mientras que esta condición no sea cierta, el bloque de sentencia se ejecuta repetitivamente, una vez la condición es cierta deja de ejecutarse el bucle.

En el siguiente ejemplo puedes ver un bucle FOR que se repite 10 veces:

```
for(int i = 0;i<10;i++)
{
        digitalWrite(VERDE, HIGH);
        delay(2000);
        digitalWrite(VERDE, LOW);
        delay(2000)
}
```

El ejemplo enciende el LED, espera dos segundos, apaga el LED y vuelve a esperar dos segundos. La condición de salida del bucle es que la variable i sea mayor o igual que 10, es decir, mientras que sea menor, el bloque de código se ejecutará y por tanto hasta que no se repita 10 veces la sentencia, se estará repitiendo el encendido y apagado del led.

Por cada repetición, la variable i va aumentando de 1 en 1 su valor, tal y como se indica con *i++*. Podríamos haber puesto *i = i + 1* o *i += 1* en lugar de *i++*, ya que el funcionamiento es el mismo.

VARIABLES

Las variables son un elemento de la programación que se utilizan para almacenar valores temporalmente. La diferencia principal con las constantes es que pueden almacenar diferentes valores que van cambiando durante la ejecución del programa.

Las variables deben de ser declaradas antes de ser utilizadas. En el siguiente ejemplo puedes ver la declaración de una variable de tipo número entero (int) y la asignación de un valor:

```
int VariableEjemplo = 0;
VariableEjemplo = 3 + 5;
```

En la primera sentencia se declara la variable (VariableEjemplo) y se le asigna el valor inicial de 0. Posteriormente, la variable almacenará el resultado de realizar la suma de los valores 3 + 5.

En el aprendizaje utilizaremos los siguientes tipos de variable:

- **int**: Representa un número entero de 16bits comprendido entre -32767 y 32768.

- **bool**: Tipo de variable que únicamente puede tener dos valores: TRUE o FALSE (Verdadero o Falso).

- **byte**: Valor numérico de 8bits sin decimales comprendido entre 0 y 255.

MATERIALES

A continuación, puedes encontrar el listado de materiales que vamos a utilizar en éste primer objetivo.



Ilustración 46. Materiales objetivo 1

FASE 1: INTERACTUAR CON UN LED

La fase 1 consiste en el montaje de un circuito electrónico básico basado en el encendido y apagado automático de un LED.

MONTAJE FÍSICO

El montaje físico para la realización de la fase 1 es el siguiente:

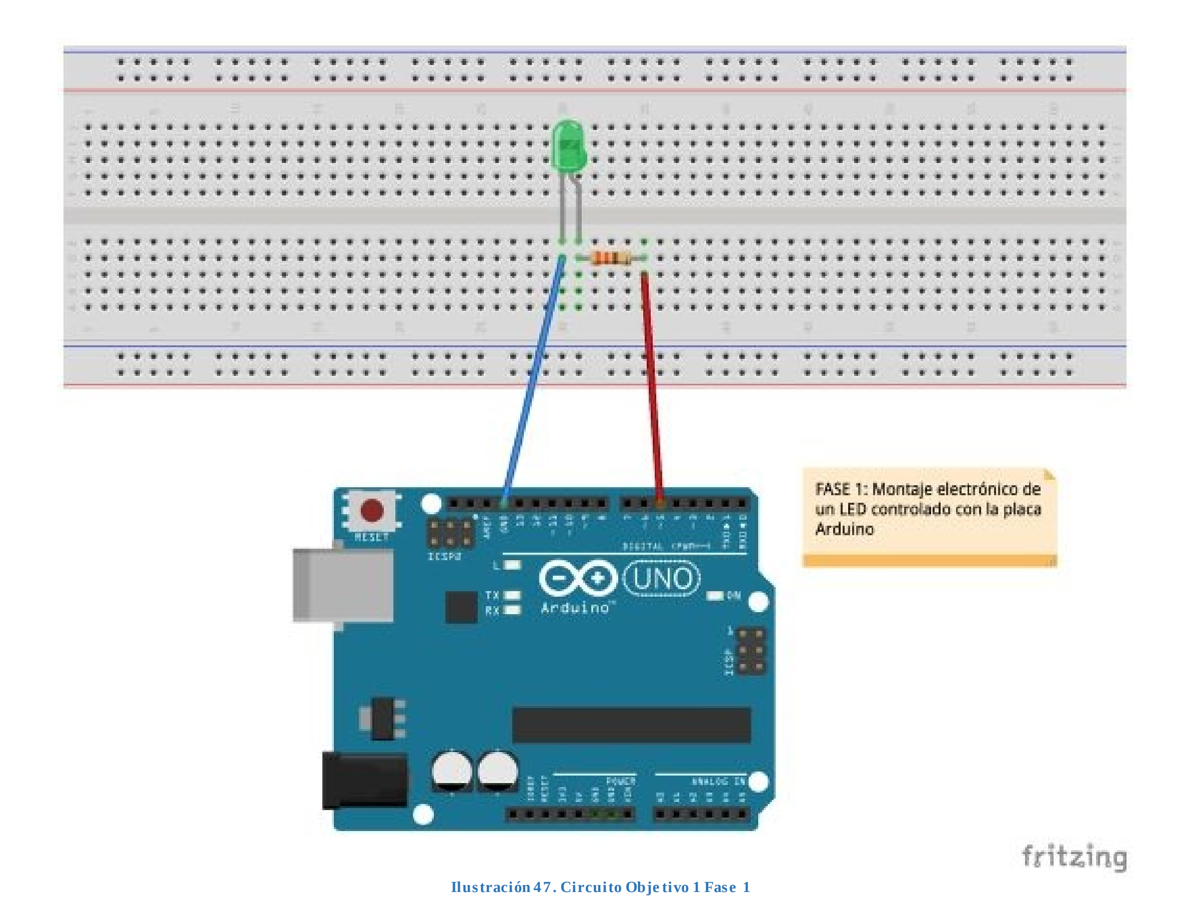


Ilustración 47. Circuito Objetivo 1 Fase 1

En el montaje de este circuito tienes que prestar especial atención a la conexión de las patillas del led. Recuerda que las resistencias son necesarias para proteger al led y que no tienen polaridad, por lo que la conexión es sencilla. Fíjate como está montado en la imagen. Si una vez montado el circuito y metido el código el led no se ilumina, revisa la conexión de las patillas por si las hubieras conectado del revés.

PROGRAMACIÓN

En la primera parte del código definiremos la constante que indicará el pin al que hemos conectado el LED. En el ejercicio hemos conectado un LED verde al pin 5, entonces el valor de dicha constante será 5. Para organizarte mejor con el código te aconsejamos que utilices nombres de constante descriptivas, no utilices nombres del estilo CONSTANTE1, CONSTANTE2…

En la sección de configuración, setup(), configuraremos el pin en modo de funcionamiento de salida y enviaremos el valor inicial. En este caso, al enviar el valor LOW le estamos estableciendo el valor de apagado al LED.

En la sección principal, loop(), encenderemos el LED, detendremos el programa un segundo, apagaremos el LED y de nuevo volveremos a detener el programa un segundo. Esta sección se ejecutará de forma indefinida, es decir, se repetirá constantemente encendiendo y apagando el LED.

CÓDIGO FUENTE

```
#define GREEN 5

void setup()
{
	pinMode(GREEN,OUTPUT);
	digitalWrite(GREEN,LOW);
}

void loop()
{
	digitalWrite(GREEN, HIGH);
	delay(1000);
	digitalWrite(GREEN, LOW);
	delay(1000);
}
```

FASE 2: INTERACTUAR CON VARIOS LEDS

La fase 2 consiste en el montaje de un circuito electrónico basado en el encendido y apagado de varios LEDs.

MONTAJE FÍSICO

A continuación, puedes ver el montaje electrónico del circuito:

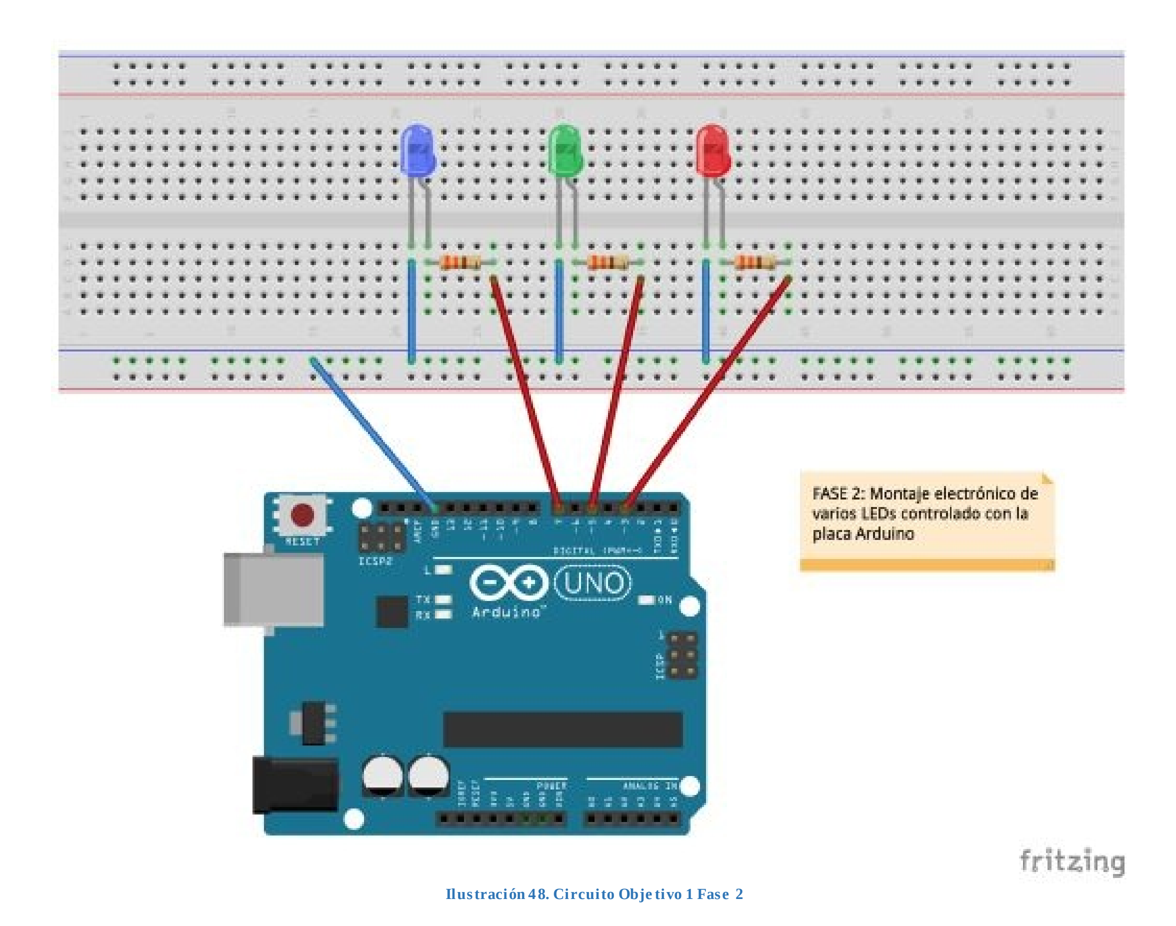


Ilustración 48. Circuito Objetivo 1 Fase 2

Igual que en el montaje anterior, prestar atención a la conexión de cada patilla de los LED. La toma de tierra ha sido conectada a la protoboard para ser compartida por todos los componentes que se utilicen, en este caso los tres LEDs.

PROGRAMACIÓN

En la primera parte del código definiremos las constantes que indican los

pines a los que hemos conectado cada uno de los tres LEDs. En el ejercicio hemos conectado el LED azul al pin número 7, el verde al pin número 5 y el rojo al pin número 3.

En la sección de configuración indicaremos los tres pines como salida, y además, estableceremos el estado inicial de todos ellos, que será el LED rojo encendido y el verde y azul apagados. El estado inicial lo establecemos con la función *digitalWrite*, escribiendo HIGH (1) en la salida del LED rojo y LOW (0) en las salidas de los LEDs azul y verde.

En la sección principal se van a ir encendiendo y apagando los LEDs, de uno en uno y deteniendo la ejecución del programa un segundo entre cada una de las operaciones que se realizan.

CÓDIGO FUENTE

```
#define BLUE 7
#define GREEN 5
#define RED 3

void setup()
{
	pinMode(RED,OUTPUT);
	pinMode(GREEN,OUTPUT);
	pinMode(BLUE, OUTPUT);
	digitalWrite(RED,HIGH);
	digitalWrite(GREEN,LOW);
	digitalWrite(BLUE, LOW);
}

void loop()
{
	digitalWrite(RED, HIGH);
	delay(1000);
	digitalWrite(RED, LOW);
	delay(1000);
	digitalWrite(BLUE, HIGH);
	delay(1000);
	digitalWrite(BLUE, LOW);
	delay(1000);
	digitalWrite(GREEN, HIGH);
	delay(1000);
	digitalWrite(GREEN, LOW);
	delay(1000);
}
```

FASE 3: INTERACTUAR CON UN LED RGB

La fase 3 y última de este primer objetivo consiste en la utilización analógica y digital de un LED RGB. El circuito que vamos a montar es compartido por ambos modos de utilización, lo único que cambia es el código fuente que cargaremos en la placa de Arduino.

MONTAJE FÍSICO

El montaje físico del circuito es el siguiente:

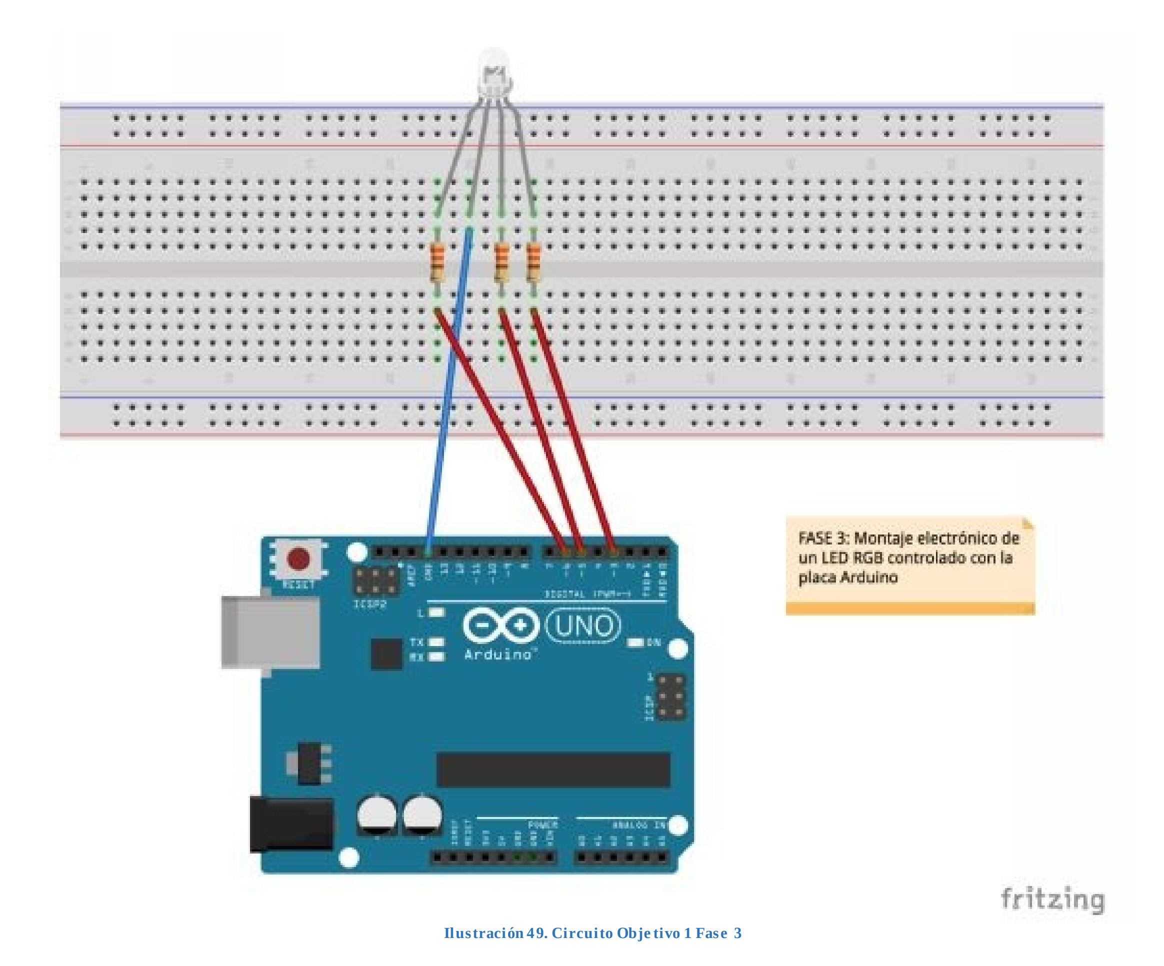


Ilustración 49. Circuito Objetivo 1 Fase 3

Tienes que prestar especial atención a la patilla del LED RGB a la que conectas la toma de tierra. Cada patilla del LED se corresponde con un color y una con la tierra, tal y como puedes ver en la siguiente imagen:

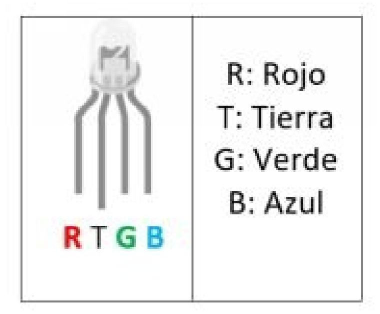


Ilustración 50. Patillas LED RGB

PROGRAMACIÓN DIGITAL

En la primera parte del código definiremos las constantes que indican los pines a los que hemos conectado cada una de los tres patillas del LED RGB que se corresponden con los colores rojo, verde y azul. En el ejercicio hemos conectado el LED azul al pin número 3, el verde al pin número 5 y el rojo al pin número 6.

En la sección de configuración especificaremos como salidas los tres pines que hemos configurado para cada uno de los colores de las patillas del LED RGB. Además, estableceremos el estado inicial de cada uno de ellos, concretamente, todas las salidas estarán inicializadas con el valor LOW.

En la sección principal vamos a ir encendiendo y apagando cada una de las patillas del LED RGB, estableciendo un pequeño *delay* para que se pueda apreciar el cambio de colores del LED RGB.

CÓDIGO FUENTE

```
#define RED 6
#define GREEN 5
#define BLUE 3

void setup()
{
	pinMode(RED, OUTPUT);
	pinMode(GREEN, OUTPUT);
	pinMode(BLUE, OUTPUT);
	digitalWrite(RED, LOW);
	digitalWrite(GREEN, LOW);
	digitalWrite(BLUE, LOW);
}
```

```
void loop()
{
	digitalWrite(RED, HIGH);
	delay(500);
	digitalWrite(RED, LOW);
	delay(500);
	digitalWrite(GREEN, HIGH);
	delay(500);
	digitalWrite(GREEN, LOW);
	delay(500);
	digitalWrite(BLUE, HIGH);
	delay(500);
	digitalWrite(BLUE, LOW);
	delay(500);
}
```

PROGRAMACIÓN ANALÓGICA

En este ejercicio utilizaremos el mismo circuito electrónico que hemos montado para el anterior. La diferencia entre ambos ejercicios reside en que en el anterior hemos utilizado las salidas de forma digital y en éste de forma analógica.

Para este ejercicio hemos añadido una nueva constante, *delayTime*, que indica el tiempo que transcurre entre cada cambio que haremos en la intensidad del LED.

La sección de configuración es exactamente igual que en el anterior ejercicio.

El cambio respecto al ejercicio anterior radica en la sección principal (*loop*), iremos encendiendo cada una de las patillas del LED RGB de forma progresiva para después apagarla con un *digitalWrite* y continuar con la siguiente patilla.

CÓDIGO FUENTE

```
#define RED 6
#define GREEN 5
#define BLUE 3
#define delayTime 20

void setup()
{
	pinMode(RED, OUTPUT);
```

```
        pinMode(GREEN, OUTPUT);
        pinMode(BLUE, OUTPUT);
        digitalWrite(RED, LOW);
        digitalWrite(GREEN, LOW);
        digitalWrite(BLUE, LOW);
}

void loop()
{
                delay(1000);

                for(int i = 0; i < 255; i += 1)
                {
                                analogWrite(GREEN, i);
                                delay(delayTime);
                }

                digitalWrite(GREEN, LOW);
                delay(1000);

                for(int i = 0; i < 255; i += 1)
                {
                                analogWrite(RED, i);
                                delay(delayTime);
                }

                digitalWrite(RED, LOW);
                delay(1000);

                for(int i = 0; i < 255; i += 1)
                {
                                analogWrite(BLUE, i);
                                delay(delayTime);
                }
                digitalWrite(BLUE, LOW);
}
```

AHORA ERES CAPAZ DE…

Has acabado el primer objetivo y has adquirido los siguientes conocimientos:

- Familiarización con el entorno de desarrollo.
- Familiarización con la placa Arduino.
- Realización de pequeños programas.
- Utilización de sentencias sencillas.
- Manejo de LEDs.
- Manejo de LED RGB.
- Montaje básico de circuitos utilizando LEDs.

EXPLICACIÓN

En el objetivo número 2 explicaremos todo lo necesario para aprender a manejar pulsadores con el fin de controlar el encendido y apagado de los LEDs.

En el objetivo anterior has aprendido a manejar las salidas de la placa. En éste, lo principal es que aprendas a interactuar con las entradas de la placa y seas capaz de combinar la interacción de ambas.

FUNCIONES DE PROGRAMACIÓN

En este objetivo vamos a profundizar en las siguientes funciones de programación:

DIGITALREAD

Mediante la sentencia digitalRead podemos leer el valor que tiene un pin almacenado. Para proceder con la lectura hay que indicar el pin a leer, que irá de 0 a 13 y podrá ser indicado mediante valor o constante. El valor leído será uno de los dos siguientes:

- HIGH: El pin contiene el valor 1.
- LOW: El pin contiene el valor 0.

En el siguiente ejemplo se recibe el valor del pin número 3:

valor = digitalRead(3);

En lugar de utilizar el valor 3, en el siguiente ejemplo te mostramos como utilizar la constante que hemos definido previamente:

valor = digitalRead(VERDE);

IF

La sentencia if se utiliza para comprobar si una condición se cumple o no y ejecutar un bloque de código u otro dependiendo de ello. Es decir, mediante if vamos a poder ejecutar diferentes sentencias dependiendo del resultado de una comparación entre dos elementos.

Las comparaciones que se pueden realizar entre los dos elementos de la comparación son:

<	Primer elemento menor que segundo elemento
<=	Primer elemento menor o igual que segundo elemento
>	Primer elemento mayor que segundo elemento
>=	Primer elemento mayor o igual que segundo elemento
!=	Primer elemento diferente a segundo elemento
==	Primer elemento igual a segundo elemento

Ilustración 51. Operadores comparación

En el siguiente ejemplo se comprueba el valor de la entrada número 5 de la placa, y en caso de ser HIGH escribe en el pin de salida el valor HIGH, es decir, si en el pin 5 tenemos una señal de entrada HIGH, en la salida se devolverá otra señal HIGH. Traducido a un ejemplo sencillo, sería: si activamos un dispositivo, provocamos la activación de otro, en este caso, encendemos el led verde.

```
If(digitalRead(5)==HIGH)
{
        digitalWrite(VERDE) = HIGH;
}
```

En la sentencia if puede añadirse el bloque de código que se debe de ejecutar en caso de que la condición no se cumpla, éste bloque es añadido con la sentencia ***else***.

En el siguiente ejemplo se comprueba si el valor del pin número 5 es HIGH y en caso de serlo escribe HIGH en la salida VERDE, por el contrario, si no se cumple, escribe HIGH en la salida ROJO.

```
  If(digitalRead(5)==HIGH)
{
        digitalWrite(VERDE) = HIGH;
}
else
  {
        digitalWrite(ROJO) = HIGH;
{
```

Traducido a un ejemplo sencillo sería: si (if) activamos un dispositivo

determinado se enciende el led verde, y si no (else) lo activamos se enciende el led rojo.

En la condición de la sentencia if también se pueden poner diferentes condiciones mediante operadores OR y AND.

- OR: Se representa por || e indica que se tiene que cumplir una u otra.
- AND: Se representa por && e indica que se tienen que cumplir una y otra.

En la siguiente sentencia se está comprobando mediante OR que la variable *valor* sea menor que *valorminimo* o que sea mayor que *valormaximo*:

if (valor < valorminimo || valor > valormaximo)

INPUT_PULLUP

Es un tipo de configuración de pin de la placa de Arduino, junto con INPUT y OUTPUT. La diferencia respecto a INPUT es que si utilizamos INPUT_PULLUP **el circuito utilizará las resistencias internas de la placa de Arduino**, cosa que no ocurre con el modo de funcionamiento INPUT.

Llevado a la práctica, si utilizamos INPUT deberemos de utilizar resistencias en nuestros circuitos para los elementos de entrada de la placa, y si utilizamos INPUT_PULLUP no tendremos que utilizarlas ya que utilizará las internas de la placa.

MATERIALES

A continuación, puedes encontrar el listado de materiales que vamos a utilizar en éste segundo objetivo:

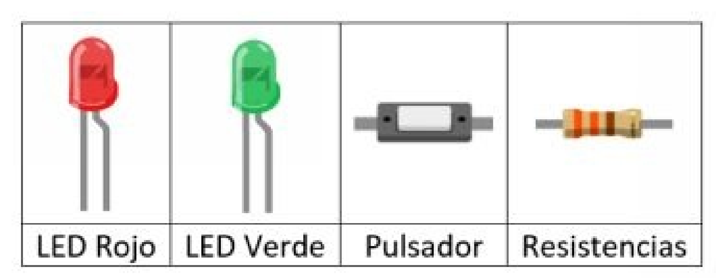


Ilustración 52. Materiales objetivo 2

FASE 1: ENCENDER Y APAGAR VARIOS LEDS CON UN PULSADOR (VERSIÓN 1)

La fase 1 de este objetivo consiste en la utilización de un pulsador para encender y apagar dos LEDs en función de su pulsación. El modo de funcionamiento del pin de entrada será INPUT.

MONTAJE FÍSICO

A continuación, tienes el montaje del circuito electrónico para esta fase:

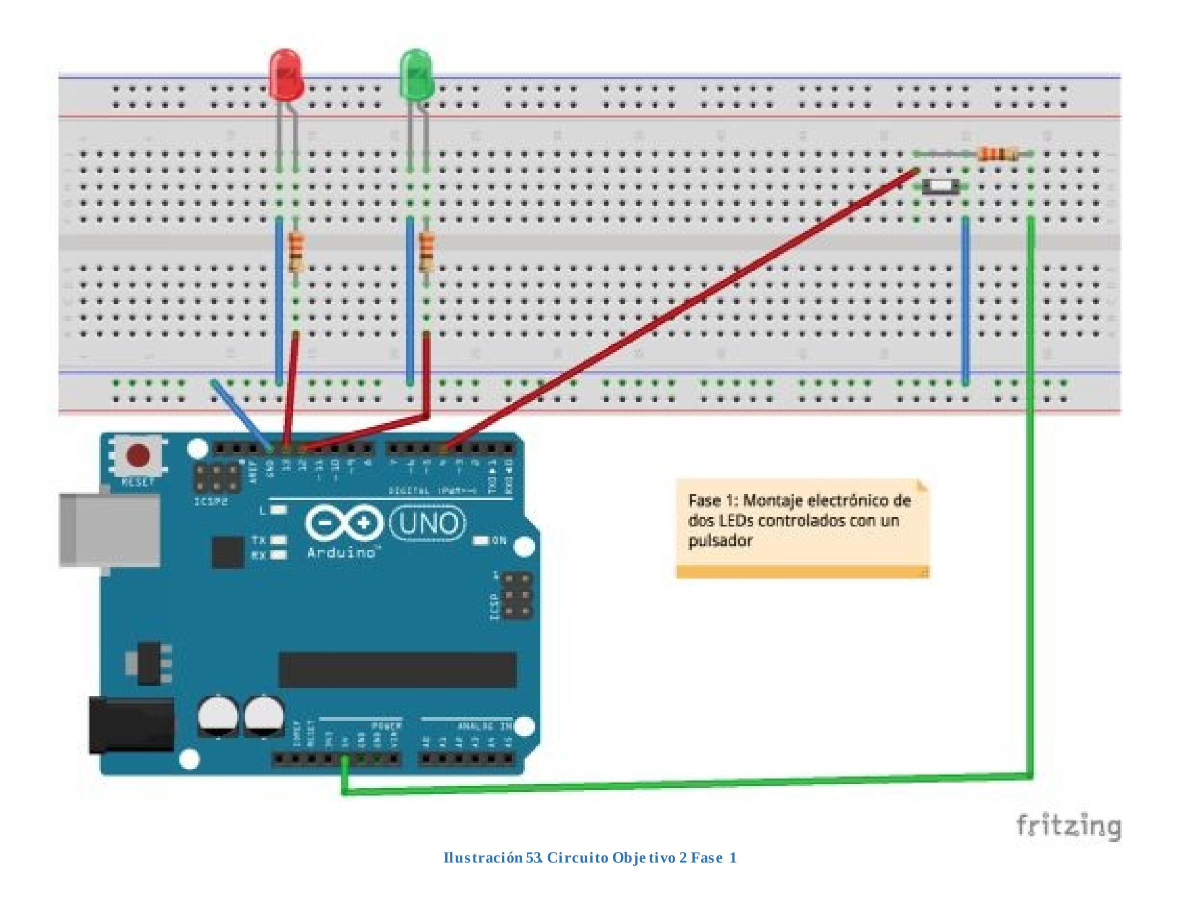


Ilustración 53. Circuito Objetivo 2 Fase 1

Tienes que prestar atención a que el pulsador esté correctamente conectado para poder leer la entrada sin problemas.

PROGRAMACIÓN

En la primera sección del programa se definen las constantes que utilizaremos en la actividad. Hemos definido para el LED rojo la constante 13 y para el

verde la 12, que son los pines donde están conectados a la placa. Para el pulsador hemos definido la constante 4, que es el pin donde lo hemos conectado.

En la sección de configuración se definen como salidas los pines donde están conectados los LEDs (la salida se refiere al encendido/apagado de los leds) y como entrada, el pin donde hemos conectado el pulsador (la entrada se refiere a la pulsación o no del pulsador).

La sección principal consiste en que si el pulsador se encuentra presionado (LOW), el LED verde se encenderá (el rojo estará apagado), en caso contrario, si no se encuentra presionado, el LED encendido será el rojo (y el verde apagado).

CÓDIGO FUENTE

```
#define RED 13
#define GREEN 12
#define BUTTON 4

void setup()
{
	pinMode(RED,OUTPUT);
	pinMode(GREEN,OUTPUT);
	pinMode(BUTTON,INPUT);
}

void loop()
{
	if(digitalRead(BUTTON)==LOW)
	{
		digitalWrite(GREEN,LOW);
		digitalWrite(RED,HIGH);
	}
	else
	{
		digitalWrite(RED,LOW);
		digitalWrite(GREEN,HIGH);
	}
}
```

FASE 2: ENCENDER Y APAGAR VARIOS LEDS CON UN PULSADOR (VERSIÓN 2)

La fase 2 del objetivo consiste en la utilización del modo de funcionamiento de pines **INPUT_PULLUP** y el manejo de dos LEDs en función de la pulsación de un pulsador.

MONTAJE FÍSICO

El circuito electrónico es el siguiente:

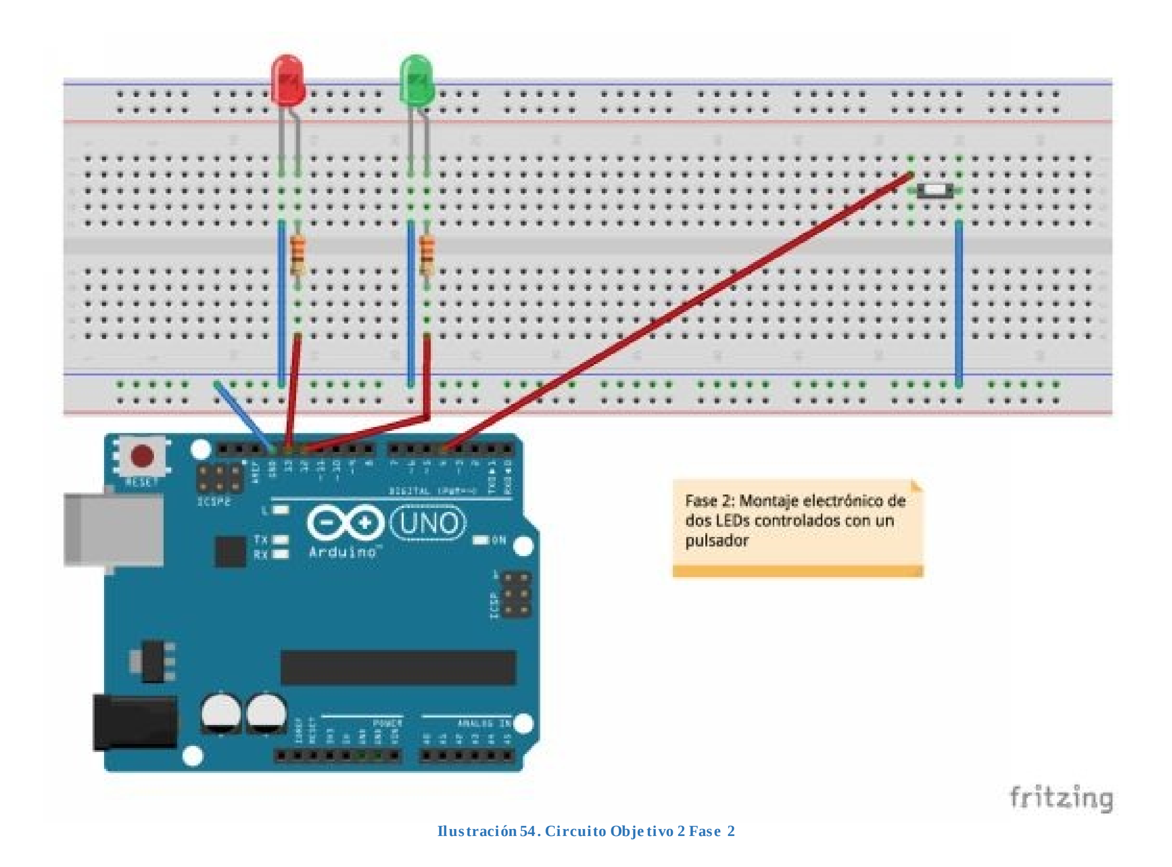


Ilustración 54. Circuito Objetivo 2 Fase 2

Debes prestar especial atención al circuito formado con el pulsador, como puedes ver, respecto al circuito de la fase anterior, hemos omitido la resistencia ya que vamos a utilizar la función **INPUT_PULLUP**.

PROGRAMACIÓN

El código fuente de este ejercicio es exactamente igual que el ejemplo anterior, a excepción del modo de configuración del pin asignado al pulsador, en el

ejercicio anterior lo definíamos como INPUT y en este ejercicio como INPUT_PULLUP.

CÓDIGO FUENTE

```
#define RED 13
#define GREEN 12
#define BUTTON 4

void setup()
{
	pinMode(RED,OUTPUT);
	pinMode(GREEN,OUTPUT);
	pinMode(BUTTON,INPUT_PULLUP);
}

void loop()
{
	if(digitalRead(BUTTON)==LOW)
	{
		digitalWrite(GREEN,LOW);
		digitalWrite(RED,HIGH);
	}
	else
	{
		digitalWrite(RED,LOW);
		digitalWrite(GREEN,HIGH);
	}
}
```

AHORA ERES CAPAZ DE…

Tras la finalización del segundo objetivo has adquirido los siguientes conocimientos:

- Has aprendido a montar un circuito con componentes de entrada a la placa.
- Has aprendido un nuevo modo de funcionamiento de pin.
- Has aprendido a leer información de entrada.
- Has ampliado las sentencias de código que sabes utilizar.

EXPLICACIÓN

En este proyecto resumen vas a aprender a simular un semáforo utilizando LEDs y un pulsador junto con funciones de programación ya conocidas en los dos objetivos anteriores.

El objetivo de la realización del semáforo es poner en práctica la aplicación de lógica de funcionamiento de elementos existentes en nuestro entorno del día a día al código fuente que escribes, junto con la consolidación de los conocimientos de construcción de circuitos y funciones de programación.

El proyecto que vas a construir simulará el funcionamiento de un semáforo convencional, la relación entre los componentes electrónicos y los del semáforo es la siguiente:

- Semáforo de cochesス Simulación con LEDs.
- Semáforo peatonesス Simulación con LEDs.
- Botón peatones para activar semáforoス Pulsador y LED de indicación de pulsación.

MATERIALES

A continuación, puedes encontrar el listado de materiales que vamos a utilizar en este proyecto:

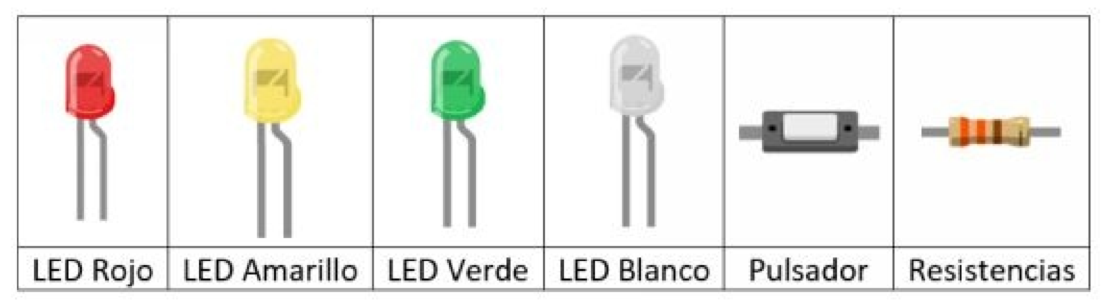


Ilustración 55. Materiales proyecto

MONTAJE FÍSICO

El circuito del proyecto es el siguiente:

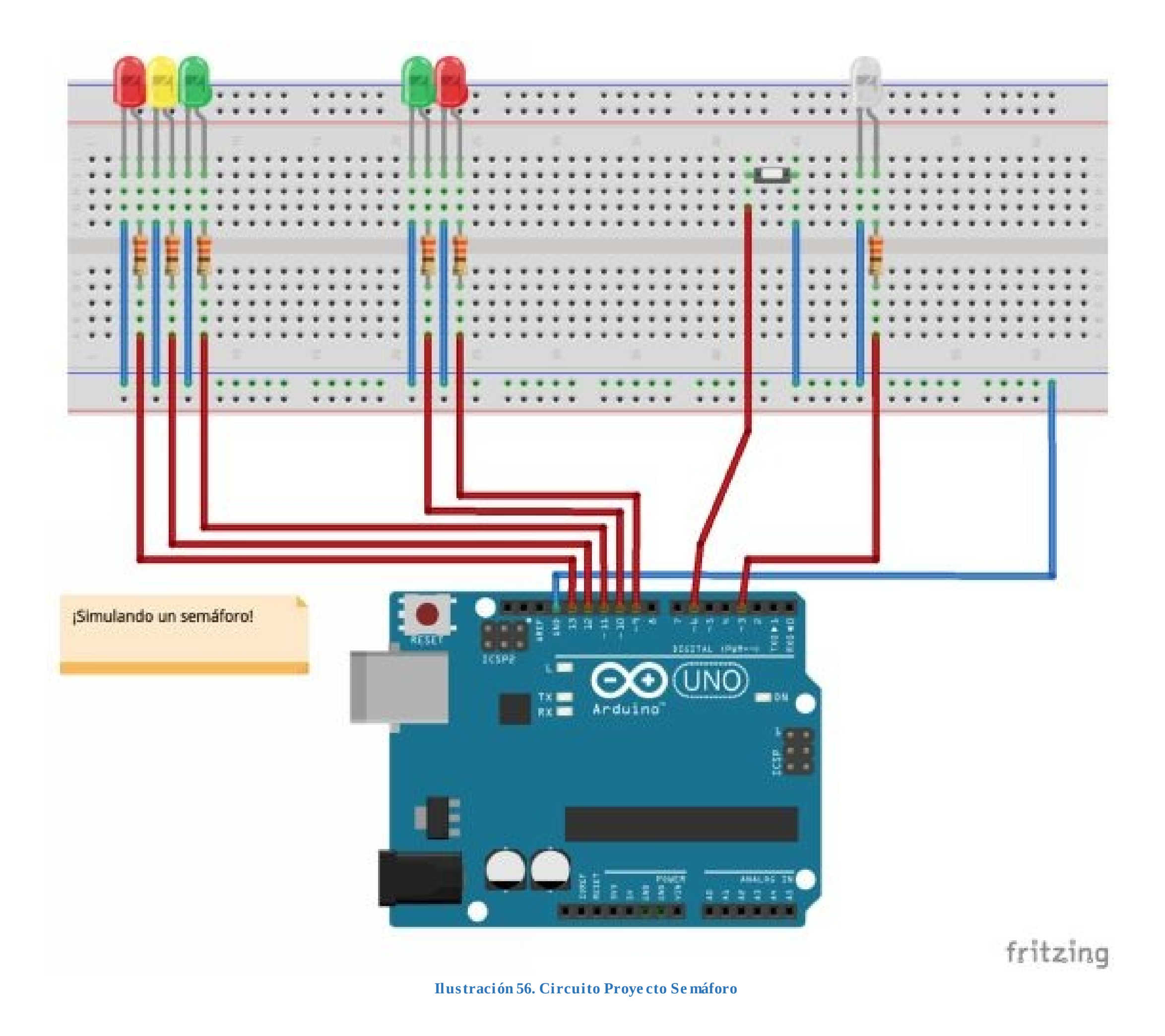


Ilustración 56. Circuito Proyecto Semáforo

En el ejemplo vas a utilizar un modo de configuración de entrada INPUT_PULLUP, por lo que no es necesario el uso de resistencias en el circuito que vas a montar con el pulsador.

PROGRAMACIÓN

La parte de definición de constantes (#define) contiene todas las definiciones de todas las entradas y salidas que vas a utilizar de la placa de Arduino. Vas a tener la definición de los 3 colores existentes en un semáforo de coches y los 2 colores existentes en el semáforo de los peatones. También la luz de indicación de que ha sido presionado el pulsador por un peatón para que el semáforo de coches se ponga en rojo y así poder pasar. Por último tienes la definición del pin al que está conectado el pulsador.

En la siguiente imagen puedes ver a qué color corresponde cada una de las constantes definidas en el código:



Ilustración 57. Relación constantes-semáforo real

En la sección de configuración se define el modo de funcionamiento de todos los pines junto con el estado inicial del circuito.

El programa principal consiste en una serie de acciones que se ejecutan una vez es presionado el pulsador y que simulan el funcionamiento de un semáforo.

CÓDIGO FUENTE

```
#define REDCARS 13
#define YELLOWCARS 12
#define GREENCARS 11
#define GREENPEDESTRIANS 10
#define REDPEDESTRIANS 9
#define SIGNAL 3
#define BUTTON 6
```

```
void setup()
{
	pinMode(REDCARS,OUTPUT);
	pinMode(YELLOWCARS,OUTPUT);
	pinMode(GREENCARS,OUTPUT);
	pinMode(REDPEDESTRIANS,OUTPUT);
	pinMode(GREENPEDESTRIANS,OUTPUT);
	pinMode(SIGNAL,OUTPUT);
	pinMode(BUTTON,INPUT_PULLUP);

	digitalWrite(REDCARS,LOW);
	digitalWrite(YELLOWCARS,LOW);
	digitalWrite(GREENCARS,HIGH);
	digitalWrite(REDPEDESTRIANS,HIGH);
	digitalWrite(GREENPEDESTRIANS,LOW);
	digitalWrite(SIGNAL,LOW);
}

void loop()
{
	if(digitalRead(BUTTON)==LOW)
	{
			digitalWrite(SIGNAL,HIGH);
			delay(5000);
			digitalWrite(YELLOWCARS,HIGH);
			digitalWrite(GREENCARS,LOW);
			delay(2000);
			digitalWrite(YELLOWCARS,LOW);
			digitalWrite(REDCARS, HIGH);
			digitalWrite(SIGNAL,LOW);
			digitalWrite(REDPEDESTRIANS,LOW);
			digitalWrite(GREENPEDESTRIANS,HIGH);
			delay(10000);
			for(int i = 0;i<3;i++)
			{
					digitalWrite(GREENPEDESTRIANS,LOW);
					delay(1000);
					digitalWrite(GREENPEDESTRIANS,HIGH);
					delay(1000);
			}
			digitalWrite(GREENPEDESTRIANS,LOW);
			digitalWrite(REDPEDESTRIANS,HIGH);
			digitalWrite(REDCARS, LOW);
			digitalWrite(GREENCARS,HIGH);
	}
}
```

La traducción del código es la correspondiente al funcionamiento de dos semáforos, uno de vehículos y otro de peatones, que podemos encontrar en cualquier calle:

En la primera parte el LED rojo del semáforo de los coches está apagado *digitalWrite(REDCARS,LOW)* y el verde encendido *digitalWrite(GREENCARS,HIGH)* (están circulando los coches) y al contrario en el caso del semáforo de los peatones, el verde apagado *digitalWrite(GREENPEDESTRIANS,LOW)* y el rojo encendido *digitalWrite(REDPEDESTRIANS,HIGH)* (los peatones no pueden cruzar).

Si se presiona el pulsador *if(digitalRead(BUTTON)==LOW)*, se ilumina el LED blanco *digitalWrite(SIGNAL,HIGH)*, indicando que lo hemos activado. Pasados 5 segundos *delay(5000)*, se ilumina el amarillo de los coches *digitalWrite(YELLOWCARS,HIGH)* para avisar que deben ir parando y a la vez se apaga el verde *digitalWrite(GREENCARS,LOW)*. Pasados 2 segundos *delay(2000)* el amarillo se apaga *digitalWrite(YELLOWCARS,LOW)* y se ilumina el rojo *digitalWrite(REDCARS, HIGH)* para que los vehículos paren totalmente. En ese momento se apaga el LED blanco *digitalWrite(SIGNAL,LOW)* y el rojo de los peatones se apaga *digitalWrite(REDPEDESTRIANS,LOW)* para iluminarse el verde *digitalWrite(GREENPEDESTRIANS,HIGH)* y así permitir el paso a los peatones.

Pasados 10 segundos *delay(10000)* el LED verde de los peatones se enciende y se apaga 3 veces durante 1 segundo (1 segundo encendido, 1 pagado y así tres veces) *for(int i = 0;i<3;i++)* avisando a los peatones que deben ir dejando de cruzar. Una vez repetido tres veces, el LED verde de los peatones se apagará *digitalWrite(GREENPEDESTRIANS,LOW)* encendiéndose el rojo *digitalWrite(REDPEDESTRIANS,HIGH)* y a la vez, el verde de los coches se encenderá *digitalWrite(GREENCARS,HIGH)* y el rojo se apagará *digitalWrite(REDCARS, LOW)*.

Aquí terminará el código que hemos creado hasta que se vuelva a presionar el pulsador que volverá a iniciar el bucle.

EXPLICACIÓN

El objetivo 3 se basa en la utilización de componentes analógicos de la placa y en utilizar nuevos componentes de salida.

Se trata de una extensión del objetivo 2, en el que has aprendido a leer componentes digitales de entradas.

FUNCIONES DE PROGRAMACIÓN

En este objetivo vamos a profundizar en las siguientes funciones de programación:

ANALOGREAD

La función nos permite la lectura analógica de las entradas 0 a la 5 (ANALOG IN: A0, A1, A2, A3, A4, A5). La lectura viene dada en un valor entre 0 y 1023.

Una característica de los pines analógicos es que no necesitan ser declarados como entradas o salidas, son siempre **pines de entrada** a la placa de Arduino.

En el siguiente ejemplo se muestra como se lee el pin número 3:

int Valor = analogRead(3);

También es posible utilizar las constantes definidas para realizar las lecturas analógicas:

int Valor = analogRead(VERDE);

ANALOGWRITE

La función nos permite escribir valores analógicos comprendidos entre 0 y 255 a los pines 3, 5, 6, 9, 10 y 11 de la placa de Arduino, que son los únicos que permiten valores analógicos. Recuerda que estos pines llevan el símbolo: ~ delante del número de pin: ~3, ~5..

Tal y como puedes ver en el siguiente ejemplo, la función requiere del número del pin al que se va a enviar el valor y el valor a enviar:

analogWrite(3, 200);

También se puede utilizan una constante para el pin:

analogWrite(VERDE,200);

El valor enviado puede ser especificado mediante una constante en vez de mediante un valor directamente:

analogWrite(VERDE, VALORMAXIMO);

MAP

La función map nos permite relacionar un valor que se encuentra en un rango de valores a otro rango de valores. En el ejemplo que estamos trabajando, la placa hace una lectura del pin analógico (potenciómetro) comprendida entre los valores 0 y 1023 y tendrá que transformar ese valor en un rango comprendido entre 0 y 255 para escribirlo en una salida (zumbador). Para realizar esta conversión usamos la función map.

En el siguiente ejemplo se muestra la utilización de la función map para convertir el valor de una variable que se encuentra en el rango 0-1023 al rango 0-255:

int ValorFinal = map(ValorInicial,0,1023,0,255);

Por lo tanto, la función se utiliza de la siguiente forma:

map(Valor a transformar, InicialMenor, InicialMayor, FinalMenor, FinalMayor);

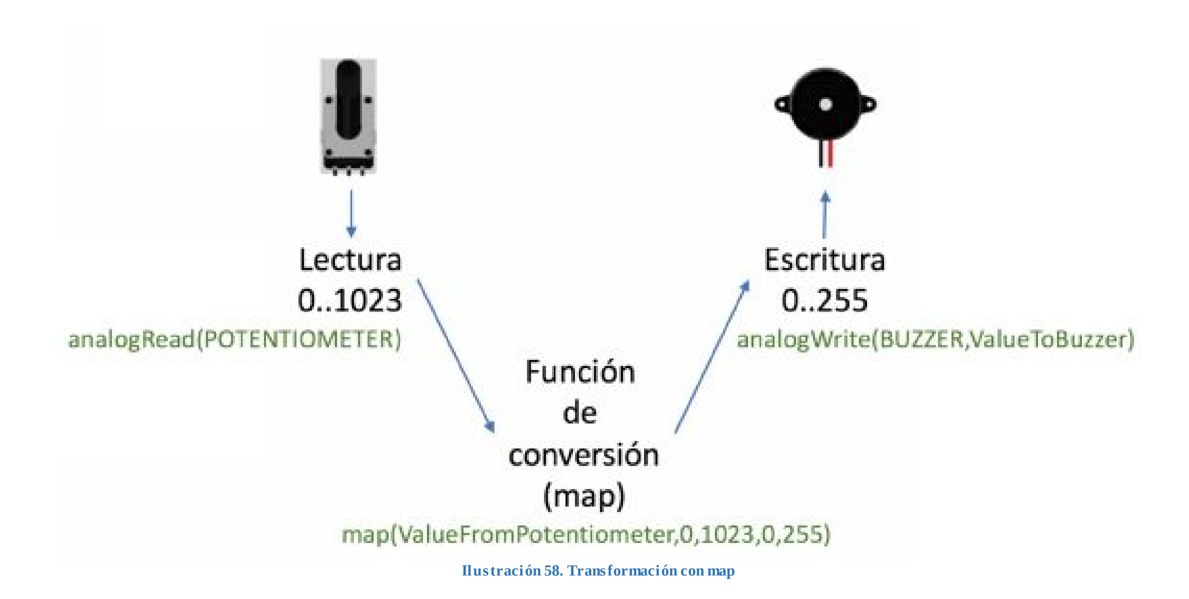


Ilustración 58. Transformación con map

MATERIALES

A continuación, puedes encontrar el listado de materiales que vamos a utilizar en este tercer objetivo:

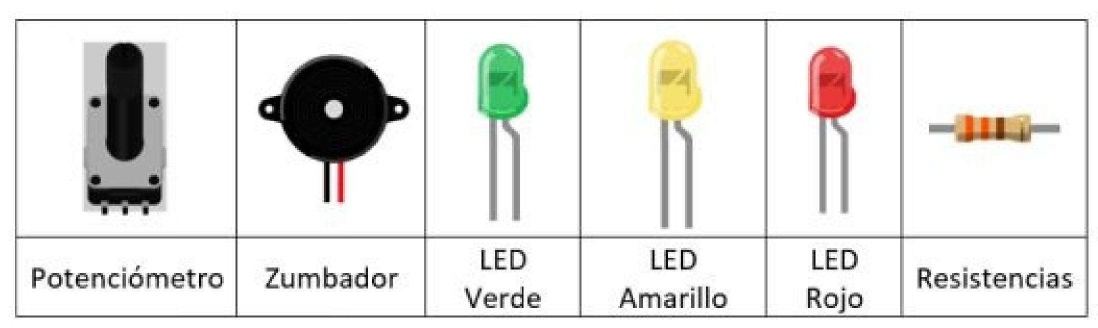


Ilustración 59. Materiales objetivo 3

FASE 1: CONTROL DE SONIDO DE UN ZUMBADOR CON UN POTENCIÓMETRO

En la fase 1 de este tercer objetivo vas a aprender a utilizar dos nuevos componentes electrónicos: un potenciómetro y un zumbador. Mediante el primero de ellos controlaremos el segundo de forma totalmente analógica: en función de la corriente que deje pasar el potenciómetro, girándolo más o menos, el zumbador emitirá un sonido más o menos elevado.

MONTAJE FÍSICO

A continuación, puedes encontrar el montaje del circuito electrónico.

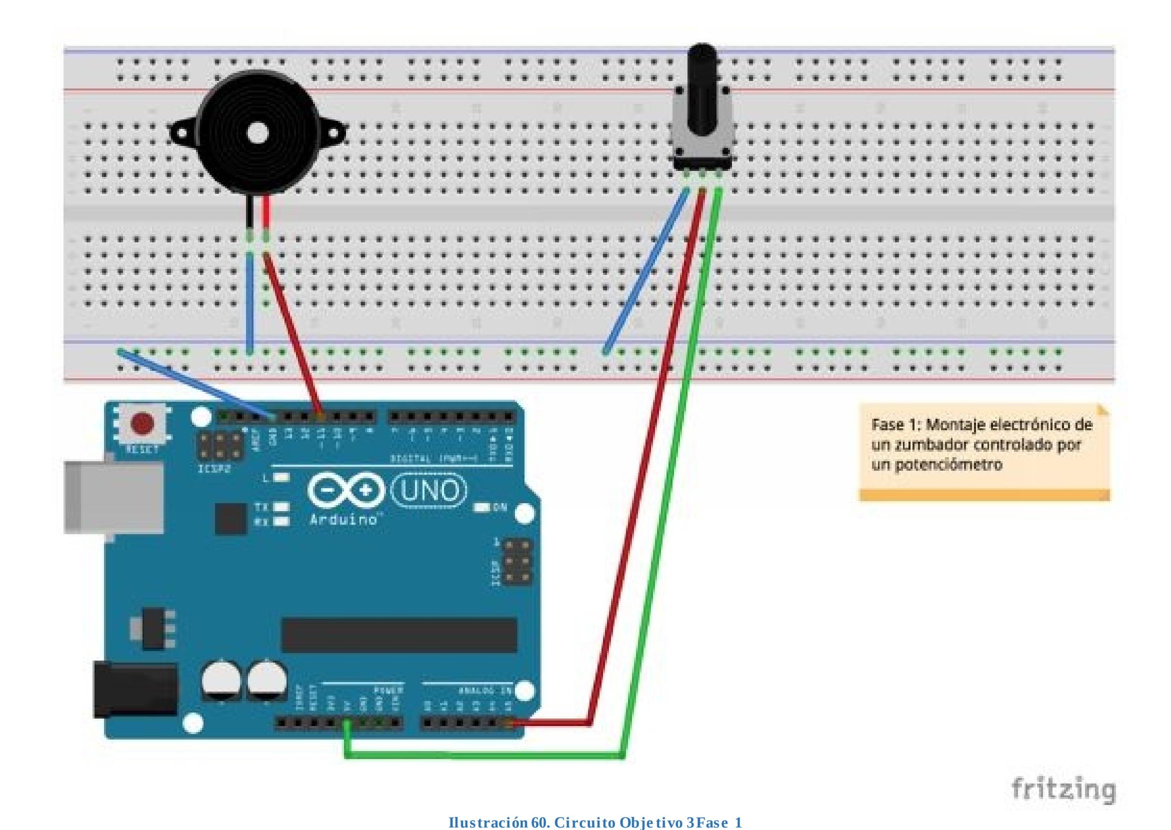


Ilustración 60. Circuito Objetivo 3 Fase 1

El zumbador conecta a tierra (GND) y a una salida analógica. El potenciómetro conecta a tierra (GND), una entrada analógica y a los 5V de la placa.

PROGRAMACIÓN

En la sección de definición hemos definidos dos constantes: la 11 para especificar el pin en el que está conectado el zumbador y la 5 para el pin donde se encuentra conectado el potenciómetro.

En la sección de configuración se define el pin del zumbador como salida *pinMode(BUZZER,OUTPUT)* y el del potenciómetro como entrada *pinMode(POTENTIOMETER,INPUT)*.

El funcionamiento de la sección principal es simple: lee el valor del potenciómetro *int ValueFromPotentiometer = analogRead(POTENTIOMETER)*, lo transforma en el rango de 0 a 255 con el que trabajan las salidas analógicas *int ValueToBuzzer = map(ValueFromPotentiometer,0,1023,0,255)* y lo escribe en la salida del zumbador *analogWrite(BUZZER,ValueToBuzzer)* para regular el volumen con el que suena .

CÓDIGO FUENTE

```
#define BUZZER 11
#define POTENTIOMETER 5

void setup()
{
	pinMode(BUZZER,OUTPUT);
	pinMode(POTENTIOMETER,INPUT);
}

void loop()
{
	int ValueFromPotentiometer = analogRead(POTENTIOMETER);
	int ValueToBuzzer = map(ValueFromPotentiometer,0,1023,0,255);
	analogWrite(BUZZER,ValueToBuzzer);
}
```

FASE 2: CONTROL DE ENCENDIDO DE LEDS CON UN POTENCIÓMETRO

En la fase 2 del objetivo vamos a manejar el encendido y apagado de LEDs mediante un potenciómetro, es decir, mediante una entrada analógica (señal enviada por el potenciómetro) vamos a controlar un conjunto de salidas digitales (señal de encendido o apagado de los LEDs).

Una vez montado y subido el código, veremos que girando más o menos el potenciómetro iremos encendiendo progresivamente los LEDs. Cuanto más giremos el potenciómetro, más LEDs iremos encendiendo y si lo giramos en sentido contrario los iremos apagando.

MONTAJE FÍSICO

El circuito electrónico es el siguiente:

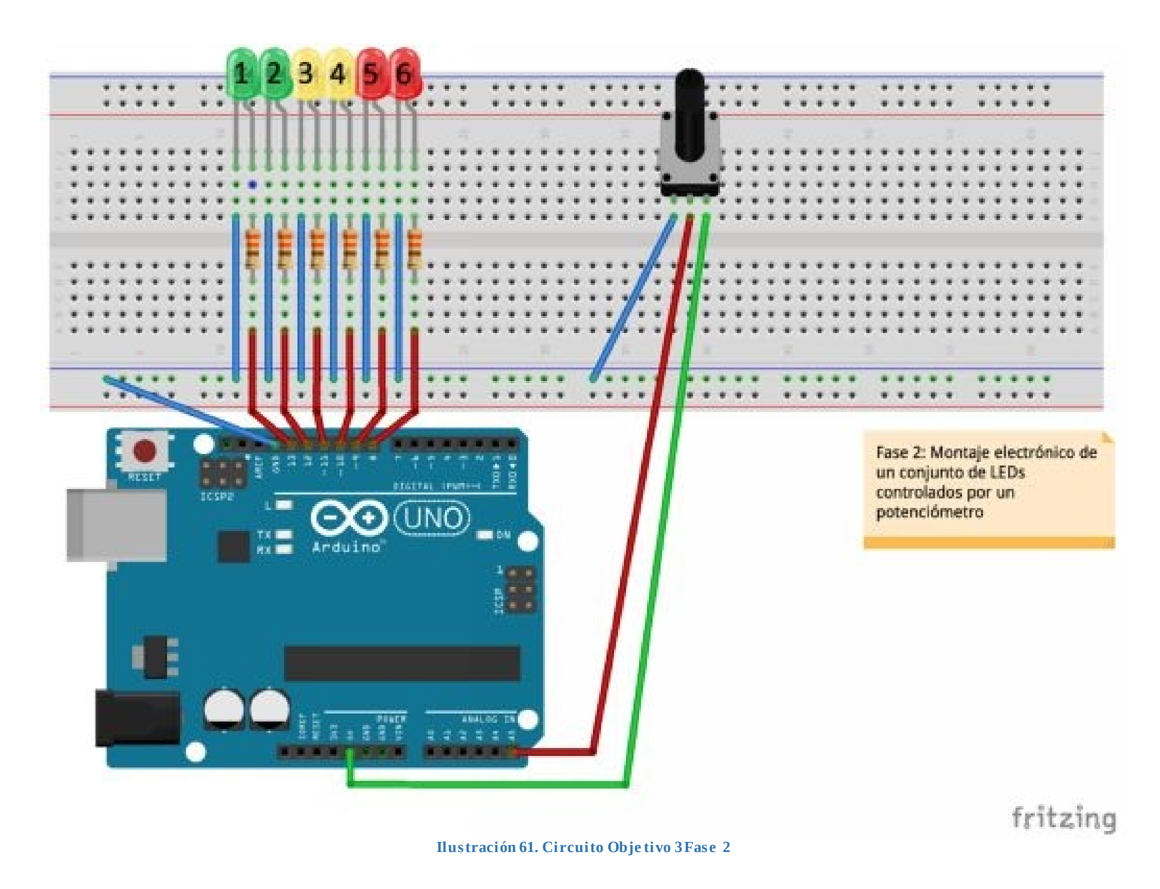


Ilustración 61. Circuito Objetivo 3 Fase 2

La conexión de los LEDs puedes verlo en el apartado del Objetivo 1 y la del potenciómetro en el apartado anterior.

PROGRAMACIÓN

En la sección de constantes están definidas todas las constantes que especifican los LEDs conectados a la placa, estos pines van del 8 al 13. A su vez se ha definido una constante para indicar el pin en el que se ha conectado el potenciómetro, el número 5.

La sección de configuración inicializa los pines de los LEDs como salida y del potenciómetro como entrada.

La sección principal es tan simple como la del ejercicio anterior, aunque más extensa en lo que a cantidad de código fuente se refiere. Básicamente tiene 3 subsecciones claramente diferenciadas, en la primera se obtiene el valor del potenciómetro *int ValueFromPotentiometer = analogRead(POTENTIOMETER)*, en la segunda se traduce el valor al rango 0 y 255 *int value = map(ValueFromPotentiometer,0,1023,0,6)* y en la última parte encenderá los LEDs que se correspondan con el valor leído desde el potenciómetro:

if(value == 0) – Ningún LED encendido (todos LOW)
else if(value == 1) – 1 LED encendido, el verde d la izquierda (1)
else if(value == 2) – 2 LEDs encendidos, el 1 y el 2
else if(value == 3) – 3 LEDs encendidos, el 1, 2 y 3
else if(value == 4) – 4 LEDs encendidos, el 1, 2, 3y 4
else if(value == 5) – 5 LEDs encendidos, el 1, 2, 3, 4 y 5
else if(value == 6) – 6 LEDs encendidos, todos

CÓDIGO FUENTE

```
#define GREEN1 13
#define GREEN2 12
#define YELLOW1 11
#define YELLOW2 10
#define RED1 9
#define RED2 8
#define POTENTIOMETER 5

void setup()
{
		pinMode(GREEN1,OUTPUT);
		pinMode(GREEN2,OUTPUT);
```

```
		pinMode(YELLOW1,OUTPUT);
		pinMode(YELLOW2,OUTPUT);
		pinMode(RED1,OUTPUT);
		pinMode(RED2,OUTPUT);
		pinMode(POTENTIOMETER,INPUT);
}

void loop()
{
		int ValueFromPotentiometer = analogRead(POTENTIOMETER);
		int value = map(ValueFromPotentiometer,0,1023,0,6);

		if(value == 0)
		{
				digitalWrite(GREEN1,LOW);
				digitalWrite(GREEN2,LOW);
				digitalWrite(YELLOW1,LOW);
				digitalWrite(YELLOW2,LOW);
				digitalWrite(RED1,LOW);
				digitalWrite(RED2,LOW);
		}
		else if(value == 1)
		{
				digitalWrite(GREEN1,HIGH);
				digitalWrite(GREEN2,LOW);
				digitalWrite(YELLOW1,LOW);
				digitalWrite(YELLOW2,LOW);
				digitalWrite(RED1,LOW);
				digitalWrite(RED2,LOW);
		}
		else if(value == 2)
		{
				digitalWrite(GREEN1,HIGH);
				digitalWrite(GREEN2,HIGH);
				digitalWrite(YELLOW1,LOW);
				digitalWrite(YELLOW2,LOW);
				digitalWrite(RED1,LOW);
				digitalWrite(RED2,LOW);
		}
		else if(value == 3)
		{
				digitalWrite(GREEN1,HIGH);
				digitalWrite(GREEN2,HIGH);
				digitalWrite(YELLOW1,HIGH);
				digitalWrite(YELLOW2,LOW);
				digitalWrite(RED1,LOW);
				digitalWrite(RED2,LOW);
		}
		else if(value == 4)
		{
				digitalWrite(GREEN1,HIGH);
				digitalWrite(GREEN2,HIGH);
```

```
                digitalWrite(YELLOW1,HIGH);
                digitalWrite(YELLOW2,HIGH);
                digitalWrite(RED1,LOW);
                digitalWrite(RED2,LOW);
        }
        else if(value == 5)
        {
                digitalWrite(GREEN1,HIGH);
                digitalWrite(GREEN2,HIGH);
                digitalWrite(YELLOW1,HIGH);
                digitalWrite(YELLOW2,HIGH);
                digitalWrite(RED1,HIGH);
                digitalWrite(RED2,LOW);
        }
        else if(value == 6)
        {
                digitalWrite(GREEN1,HIGH);
                digitalWrite(GREEN2,HIGH);
                digitalWrite(YELLOW1,HIGH);
                digitalWrite(YELLOW2,HIGH);
                digitalWrite(RED1,HIGH);
                digitalWrite(RED2,HIGH);
        }
}
```

AHORA ERES CAPAZ DE…

Una vez realizado el objetivo número 3 has aprendido a:

- Utilizar entradas analógicas.
- Utilizar salidas analógicas.
- Mapear valores entre rangos diferentes.
- Has aumentado el número de componentes que sabes utilizar.

EXPLICACIÓN

El objetivo 4 consiste en aprender a utilizar diferentes sensores conectados a la placa de Arduino.

Además de los sensores, se van a utilizar nuevos componentes de salida junto con el ”Monitor Serie” de Arduino. (Ver apartado “Pantalla Principal”)

FUNCIONES DE PROGRAMACIÓN

En este objetivo vamos a profundizar en las siguientes funciones de programación:

LIBRERÍAS

Una librería es un componente software que contiene operaciones ya implementadas y que ofrecen un interfaz conocido que permiten el uso de las mismas.

Funcionalmente hablando, una librería es un fichero que añadimos a nuestro programa y que contiene código fuente ya escrito que podemos utilizar a través del interfaz que ofrecen.

Arduino ya incluye librerías en la instalación del IDE del desarrollo, pero, además, en internet podrás encontrar un montón de librerías que puedes incorporar a tus programas. Te animo a que busques porque vas a encontrar librerías que te van a poder ayudar a desarrollar tus propios programas.

#INCLUDE

Mediante la sentencia *#include* puedes incluir para el uso en tu programa software librerías ya desarrolladas.

En este objetivo se han utilizado algunas librerías para incluir funcionalidades ya desarrolladas en el código fuente. Las librerías que se han utilizado son:

- LiquidCrystal: Contiene todos los elementos necesarios para interactuar de forma sencilla con una pantalla LCD.
- SimpleDHT: Contiene todos los elementos necesarios para interactuar con el sensor de humedad y temperatura que se ha utilizado.

MATERIALES

A continuación, puedes encontrar el listado de materiales que vamos a utilizar en este cuarto objetivo:

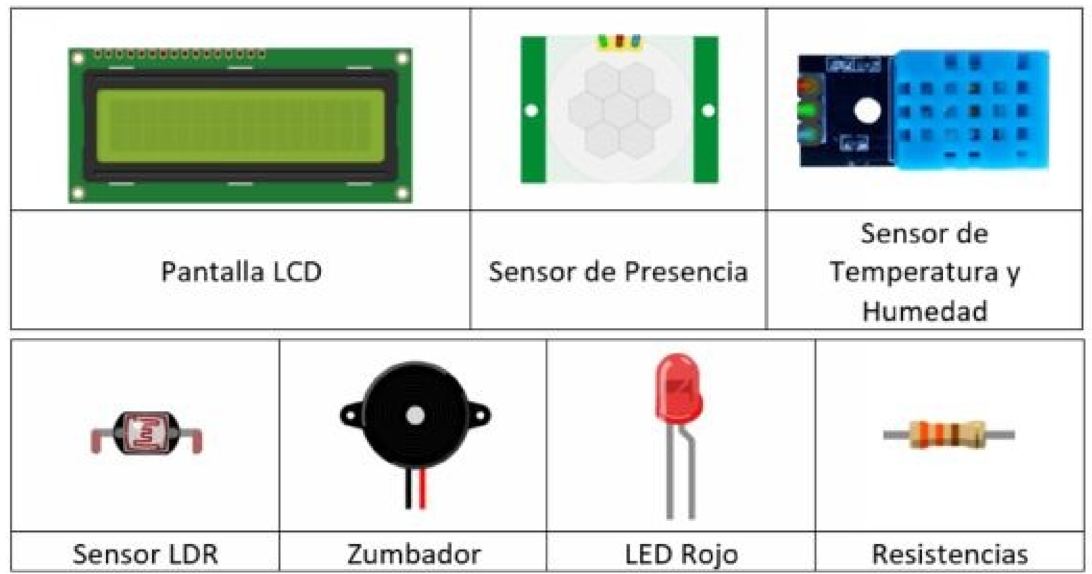


Ilustración 62. Materiales objetivo 4

FASE 1: LECTURA DE UN SENSOR LDR

En esta fase 1 del último objetivo vamos a utilizar un sensor LDR para obtener la intensidad de la luz ambiental y mostrarla en una pantalla LCD.

MONTAJE FÍSICO

El circuito electrónico es el siguiente:

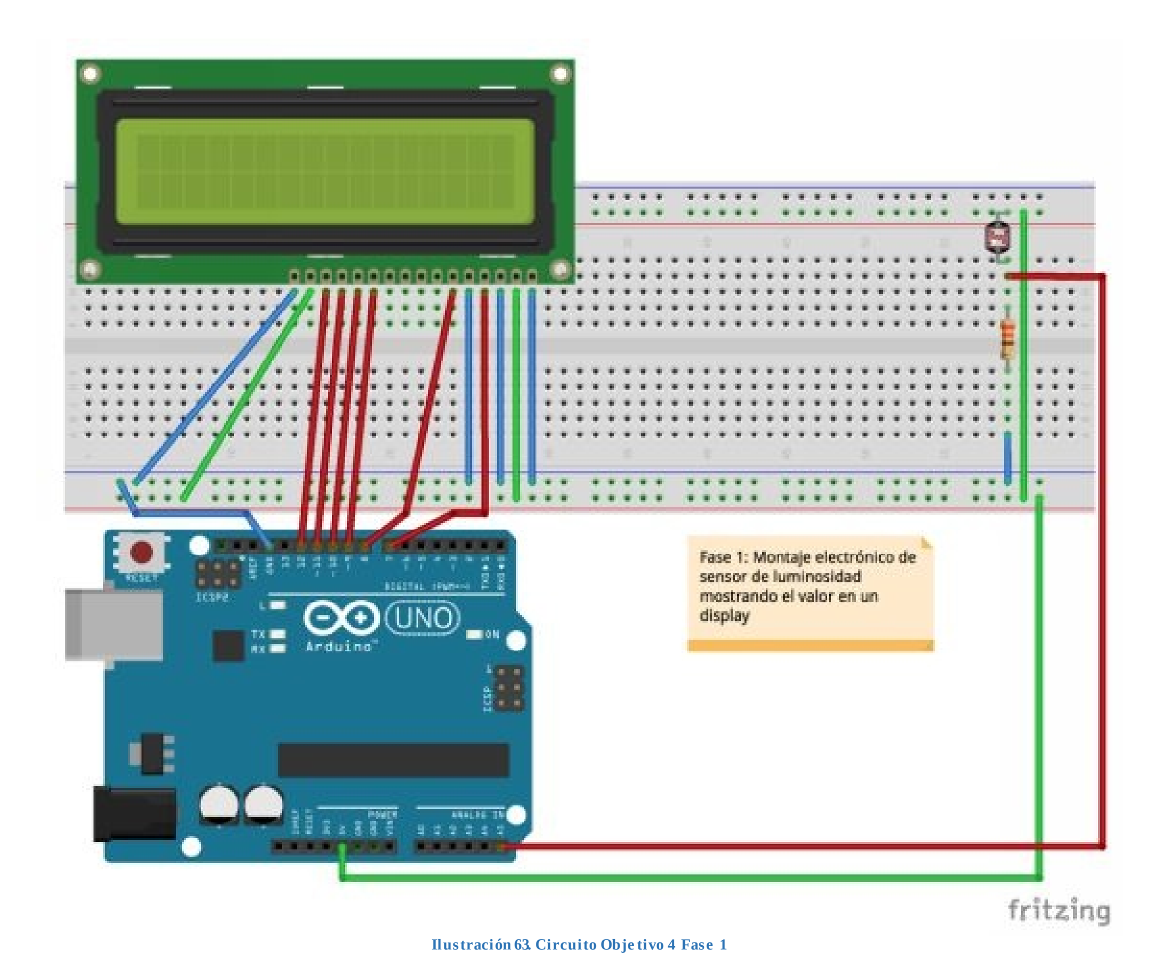


Ilustración 63. Circuito Objetivo 4 Fase 1

Debes prestar atención a las conexiones de la placa Arduino y la pantalla LCD, ya que es la primera vez que utilizarás un número tan amplio de cables para un único componente.

Revisa el apartado “PANTALLA LCD” donde podrás ver más detalles sobre las conexiones entre la pantalla y la placa Arduino.

PROGRAMACIÓN

En la sección de definiciones se definen constantes y variables, además de incluir la librería necesaria para interactuar de forma fácil y rápida con la pantalla LCD que hemos conectado a la placa *#include <LiquidCrystal.h>*. Las constantes definidas indican que el sensor LDR está conectado al pin 5 mientras que la pantalla LCD está conectada a los pines que van del 7 al 12, ambos inclusive. En esta sección también se definen las variables para la pantalla LCD y un par de variables que se utilizarán en la sección principal.

La sección de configuración inicializa los diferentes pines como entrada o salida según corresponda. Además, de las dos líneas de la LCD en las que se mostrará información, en la fila superior se mostrará una breve descripción de lo que se mostrará en la segunda línea, "Nivel de luz".

La sección principal lee el valor del sensor LDR y lo transforma a un valor de 0 a 100, ya que se mostrará el nivel de luz de forma porcentual. Una vez se tiene el valor transformado, éste es escrito en la segunda línea del LCD.

CÓDIGO FUENTE

```
#include <LiquidCrystal.h>
#define LDR 5
#define PIN1 7
#define PIN2 8
#define PIN3 9
#define PIN4 10
#define PIN5 11
#define PIN6 12

LiquidCrystal lcd(PIN1,PIN2,PIN3,PIN4,PIN5,PIN6);
int value, normalizedValue;

void setup()
{
	pinMode(LDR, INPUT);
	pinMode(PIN1, OUTPUT);
	pinMode(PIN2, OUTPUT);
	pinMode(PIN3, OUTPUT);
	pinMode(PIN4, OUTPUT);
	pinMode(PIN5, OUTPUT);
	pinMode(PIN6, OUTPUT);
	lcd.begin(16, 2);
	lcd.print("Nivel de luz");
}
```

```
void loop()
{
	value = analogRead(LDR);
	normalizedValue = map(value,0,1023,0,100);
	lcd.setCursor(0, 1);
	lcd.print(normalizedValue);
	lcd.print("%");
}
```

FASE 2: SEGURIDAD CON SENSOR DE PRESENCIA

La fase 2 consiste en la simulación de una pequeña alarma que se activa con el sensor de presencia y movimiento. Si el sensor detecta que existe movimiento o presencia activará el zumbador, emitiendo una señal acústica, y unos LEDs, emitiendo una señal luminosa.

MONTAJE FÍSICO

A continuación, puedes observar el circuito electrónico:

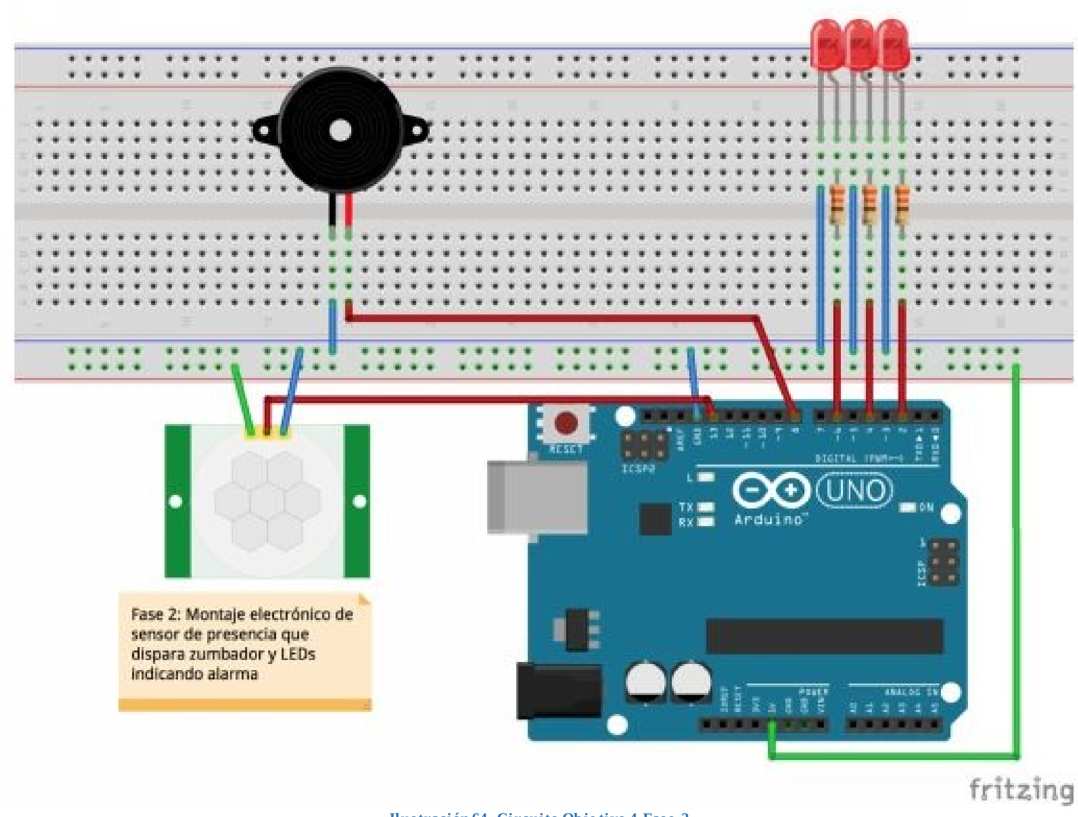


Ilustración 64. Circuito Objetivo 4 Fase 2

El circuito es bastante familiar a los que hemos tratado en todos los objetivos anteriores, con la única salvedad de la utilización del sensor de presencia y movimiento, sobre el que deberás prestar atención a la hora de conectar correctamente cada una de las conexiones.

PROGRAMACIÓN

En la sección de declaración de constantes hemos definidos cinco constantes diferentes para cada uno de los pines que vamos a utilizar. El sensor de presencia está conectado al pin número 13, el zumbador en el 8 y los tres LEDs en el 2, 4 y 6.

En la sección de configuración se define el modo de funcionamiento de cada uno de lo pines que se van a utilizar, todos ellos son pines de salida excepto el del sensor, que es un pin que se utilizará como entrada.

En el programa principal se comprobará el estado del sensor. En caso de detectar presencia *if(digitalRead(PIRSENSOR) == HIGH)*, encenderá los LEDs *digitalWrite(LED1, HIGH)* y activará el zumbador *digitalWrite(BUZZER, HIGH)*. Hemos configurado un delay de tres segundos *delay(3000)* antes de volver a comprobar el estado de zumbador, de este modo será más sencillo comprobar que todo funciona correctamente, ya que, si no introducimos este delay, la señal acústica y visual que se activa por la detección de presencia puede pasar inadvertida por nosotros. En el momento en el que el sensor deja de detectar presencia apaga los LEDs *digitalWrite(LED1, LOW)* y desactiva el zumbador *digitalWrite(BUZZER, LOW)*.

CÓDIGO FUENTE

```
#define PIRSENSOR 13
#define BUZZER 8
#define LED1 6
#define LED2 4
#define LED3 2

void setup()
{
	pinMode(BUZZER, OUTPUT);
	pinMode(LED1, OUTPUT);
	pinMode(LED2, OUTPUT);
	pinMode(LED3, OUTPUT);
	pinMode(PIRSENSOR, INPUT);
}

void loop()
{
	if(digitalRead(PIRSENSOR) == HIGH)
	{
			digitalWrite(BUZZER, HIGH);
			digitalWrite(LED1, HIGH);
			digitalWrite(LED2, HIGH);
			digitalWrite(LED3, HIGH);
			delay(3000);
```

```
        }
        else
        {
                digitalWrite(BUZZER, LOW);
                digitalWrite(LED1, LOW);
                digitalWrite(LED2, LOW);
                digitalWrite(LED3, LOW);
            }
}
```

FASE 3: LECTURA DE UN SENSOR DE TEMPERATURA Y HUMEDAD

La fase 3 es la fase más sencilla del objetivo en lo que a la construcción del circuito se refiere, ya que, lo importante de esta fase es el aprendizaje de la utilización de la consola serial de Arduino.

MONTAJE FÍSICO

El circuito electrónico que vamos a utilizar es el siguiente:

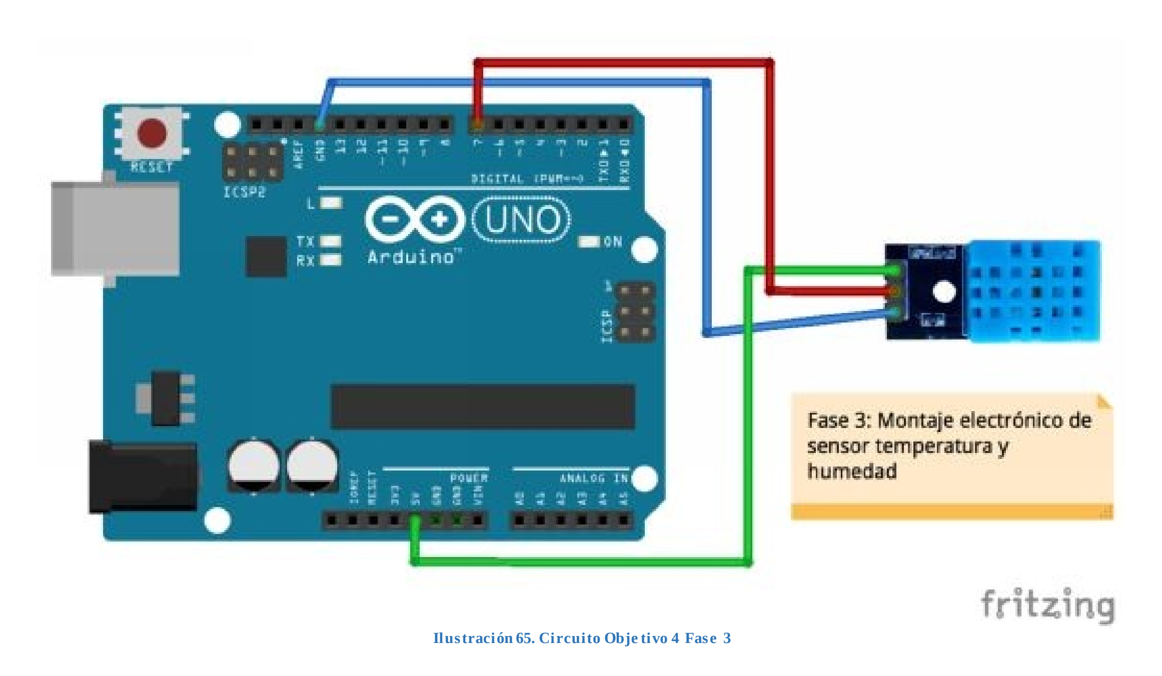


Ilustración 65. Circuito Objetivo 4 Fase 3

El montaje del circuito es muy sencillo, el sensor DHT11 tiene 3 conexiones posibles, por lo que deberás de conectar correctamente cada una de ellas al pin de la placa que corresponda. Además, en el mismo sensor (aunque no en todos) puedes ver la leyenda de cada uno de los pines a los que debe ir conectado.

PROGRAMACIÓN

La sección de configuración del ejercicio tiene tres sentencias diferentes que realizan operaciones diferentes. Mediante *#include* hemos incluido la librería que contiene todas las funciones de interacción del sensor de temperatura y humedad que hemos utilizado *#include <SimpleDHT.h>*. También hemos definido el pin 7 para el sensor *#define SENSOR 7*, y por último, hemos declarado la variable

que utilizaremos para interactuar con el sensor *SimpleDHT11 sensor*.

La sección de configuración establece el pin del sensor como entrada *pinMode(SENSOR, INPUT)* e inicializa la salida Serial *Serial.begin(9600)*, además se indica por pantalla mediante el Serial que ha sido inicializado el sensor.

Al igual que se configuran los pines de entrada y salida, al iniciar el Monitor Serie *Serial.begin(9600),* debes de configurar la velocidad de comunicación entre la placa de Arduino y el ordenador, que en este caso lo hemos configurado a 9600bps que es el valor típico de comunicación del puerto serie.

El programa principal lee los datos del sensor y los muestra por la salida del Monitor Serie. El proceso lo repite cada minuto.

La función *Serial.println("texto")* manda la información (lectura del sensor) al Puerto serie y mostrará dicha información en el Monitor Serie.

CÓDIGO FUENTE

```
#include <SimpleDHT.h>
#define SENSOR 7

SimpleDHT11 sensor;

void setup()
{
	pinMode(SENSOR, INPUT);
	Serial.begin(9600);
	Serial.println("SENSOR STARTED");
}

void loop()
{
	Serial.println("Reading the sensor...");

	byte temperature = 0;
	byte humidity = 0;
	byte data[40] = {0};
	if (sensor.read(SENSOR, &temperature, &humidity, data))
	{
			Serial.print("There was an error reading!");
	}

		Serial.println("... Sensor readed!");
		Serial.print("Temperature: ");
		Serial.print((int)temperature);
```

```
		Serial.println(" *C, ");
		Serial.print("Humidity: ");
		Serial.print((int)humidity);
		Serial.println(" %");
		Serial.println("###########");
		delay(60000);
}
```

SALIDA DE LA LECTURA (MONITOR SERIE)

Para arrancar el Monitor Serie puedes hacerlo pinchando en su icono desde la barra de accesos directos.



En la siguiente imagen puedes comprobar una ejecución del ejercicio que acabamos de realizar, en ella puedes apreciar el mensaje de inicialización del sensor y varias mediciones realizadas.

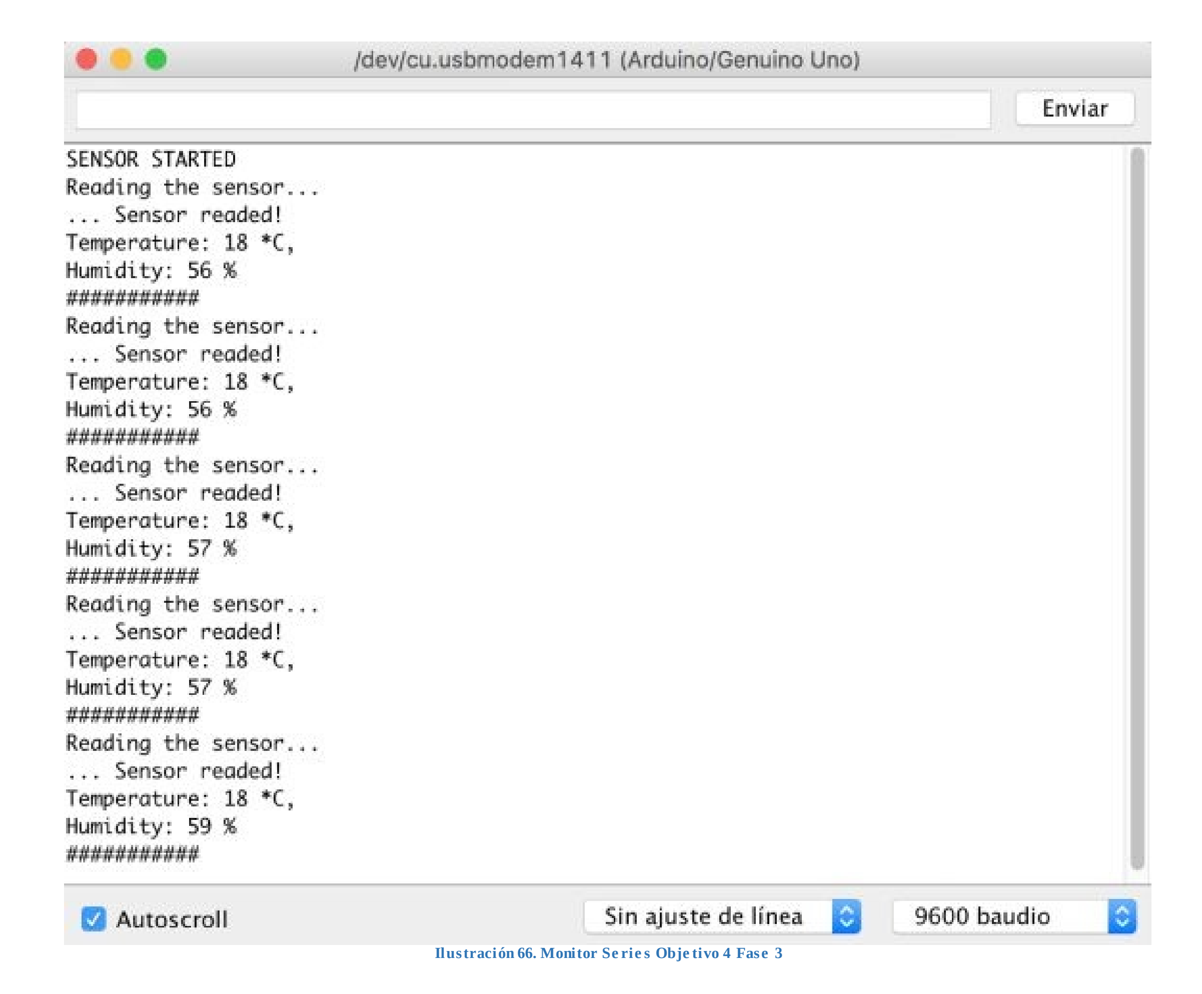


Ilustración 66. Monitor Series Objetivo 4 Fase 3

AHORA ERES CAPAZ DE…

Una vez concluido el objetivo 4, eres capaz de:

- Utilizar diferentes sensores, tanto analógicos como digitales
- Manejar la consola serial de Arduino.
- Incluir librerías externas a tus proyectos de Arduino.

EXPLICACIÓN

El proyecto final consiste en la utilización de una amplia variedad de componentes dentro de un mismo proyecto interactuando todos ellos al mismo tiempo.

Con este proyecto vamos a construir un circuito complejo que te permitirá tener controlados los parámetros ambientales más importantes de tu casa, como pueden ser la temperatura, la humedad y la luz, a la vez que manejarás sensores para detectar cualquier tipo de intrusión mediante el sensor de presencia. Además, detectaremos también cualquier tipo de fuga de agua que puedas existir mediante la utilización del sensor de detección de agua.

Para cada uno de los sensores de medición vas a establecer una serie de umbrales que te permitan definir valores mínimos y máximos para cada una de las mediciones incluidas en el proyecto.

Además, en el circuito, vas a incorporar un simulador de alarma compuesto por un zumbador y un conjunto de LEDs rojos. La alarma se activará si se detectan valores fuera de los umbrales establecidos para cada uno de los sensores de medición y también cuando se detecte presencia o fuga de agua.

El circuito incorpora también un display LCD que muestra los valores de las mediciones que está leyendo y en caso de producirse una alarma indica el sensor que la ha originado.

MATERIALES

A continuación, puedes ver todos los componentes que vamos a utilizar en este proyecto final:

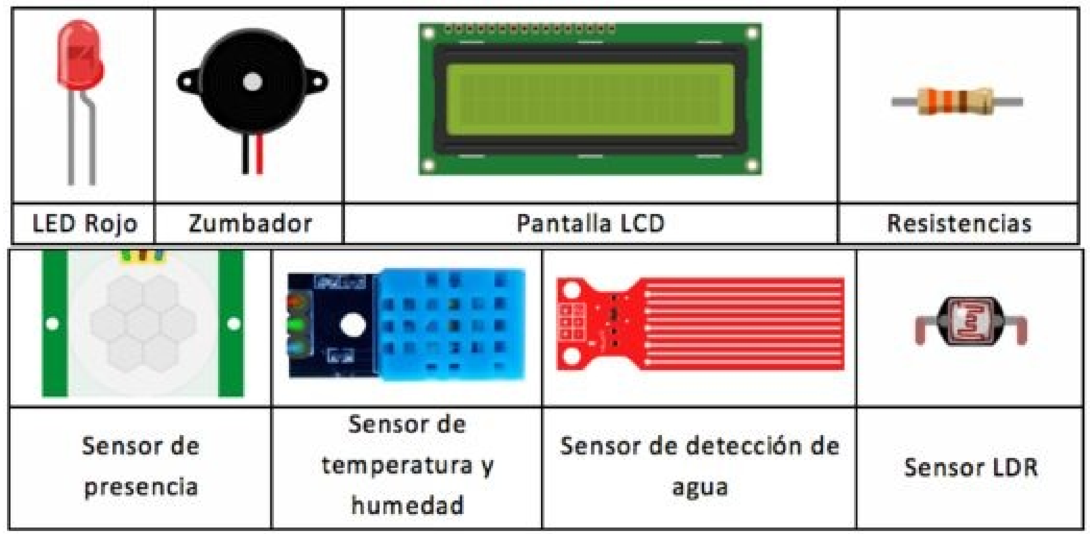


Ilustración 67. Materiales proyecto final

MONTAJE FÍSICO

El circuito del proyecto final es el siguiente:

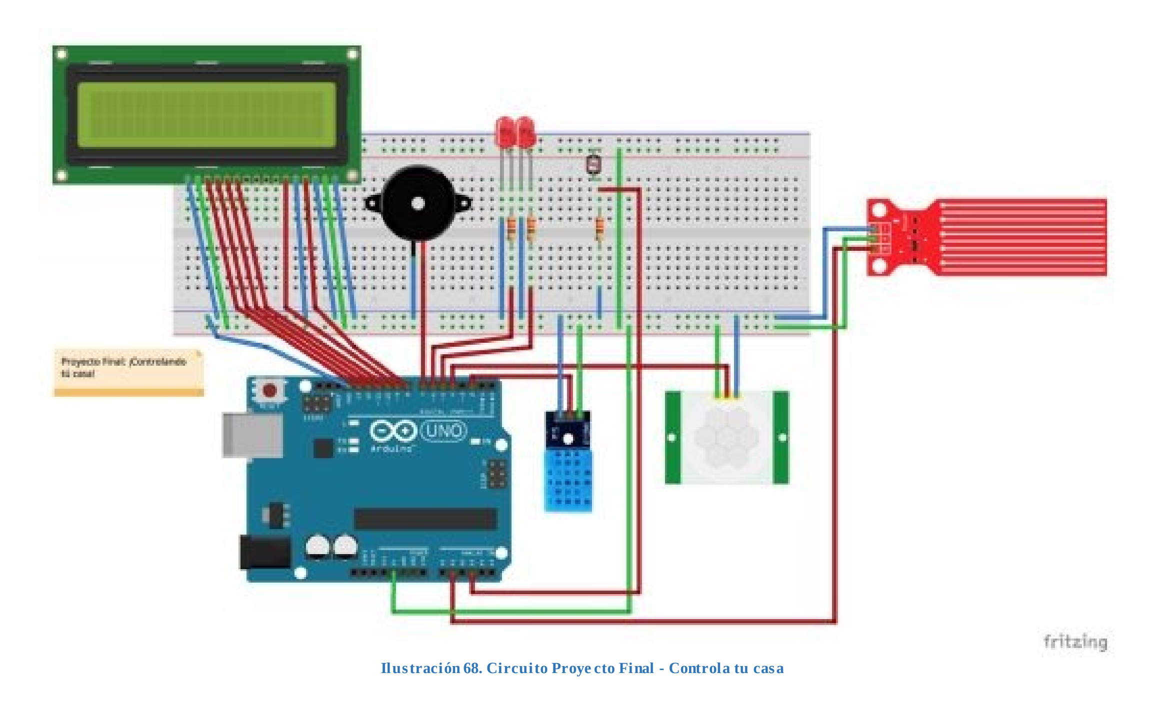


Ilustración 68. Circuito Proyecto Final - Controla tu casa

Debes prestar especial atención a las conexiones de la pantalla LCD y a las conexiones a los pines de los diferentes sensores. Tienes que tener en cuenta que las entradas analógicas tienen que estar conectadas a pines que permiten ser analógicos, no únicamente digitales. Otra cosa a tener en cuenta es que como queremos utilizar el sensor de agua en nuestra estación meteorológica para detectar si llueve o no, debemos colocar el sensor en posición horizontal para que las gotas incidan sobre toda su superficie. A medida que las gotas van cayendo sobre el sensor se formará una película de agua sobre su superficie lo que provocará un aumento del valor del pin S y por tanto detectar si está lloviendo o no.

Al haber un gran número de cables conectores diferentes sería bueno que utilizases un color para cada tipo de conexión.

Para un correcto montaje del circuito te aconsejo que vayas montándolo **poco a poco**, comprobando que funcionan todos y cada uno de los componentes que vas incluyendo. De esta forma, te asegurarás que el circuito electrónico final que montes es correcto y te resulte más fácil encontrar posibles errores de montaje.

PROGRAMACIÓN

En la primera parte del código fuente se han incluido las librerías externas que se han utilizado para interactuar con el sensor de temperatura y humedad *#include <SimpleDHT.h>* y con el display LCD *#include <LiquidCrystal.h>*. También están definidas todas las constantes en dos grupos: el primer grupo de definición contiene las constantes de los pines de la placa que utiliza cada componente *#define WATER 1*, *#define DHT 2…*, y el segundo grupo contiene los diferentes umbrales para cada uno de los sensores *#define minTemperature 10*, *#define maxTemperature 30…*, que puedes establecerles el valor que consideres conveniente para adecuar las alarmas al entorno en el que estés. Dentro de esta primera parte están incluidas las declaraciones de las variables que son utilizadas por el programa.

En la sección de configuración se inicializan cada uno de los diferentes pines de la placa que se van a utilizar junto con el modo de funcionamiento de los mismos. Además, también inicializaremos el display LCD.

La sección principal es extensa pero sencilla. Tal y como puedes ver se van comprobando cada uno de los diferentes sensores y se comprueban los valores obtenidos con sus umbrales correspondientes y se marcará ese sensor en estado de alarma en caso de que esté fuera de dichos umbrales. Una vez comprobados todos los sensores se realizará la comprobación de si alguno de ellos ha disparado una alarma. El display LCD mostrará los valores para la temperatura, humedad y luz en caso de no haberse producido ninguna alarma, en caso contrario se activarán las señales luminosas y sonoras y en el display se mostrará el mensaje de alarma junto con el código del sensor que la ha provocado. Los códigos de los sensores son los siguientes:

- T: Temperatura
- H: Humedad
- L: Luz
- P: Presencia
- W: Agua

Dada la longitud del programa, para la explicación del flujo del mismo hemos utilizado comentarios incrustados en el código fuente. Los comentarios se introducen con // o mediante bloques /* *texto* */. Las líneas de comentarios son

ignoradas durante la ejecución del programa y son utilizadas para documentar el código fuente que se escribe.

CODIGO FUENTE

```
// Incluimos las librerías que vamos a utilizar
#include <LiquidCrystal.h>
#include <SimpleDHT.h>

// Definimos las constantes de los pines de la placa, donde van conectados los diferentes sensores en la placa. Ejemplo, en el pin 1 va conectado el sensor de agua, en el 2 el sensor DHT…
#define WATER 1
#define DHT 2
#define LDR 3
#define PIRSENSOR 4
#define LED1 5
#define LED2 6
#define BUZZER 7
#define PIN1 8
#define PIN2 9
#define PIN3 10
#define PIN4 11
#define PIN5 12
#define PIN6 13

// Definimos los umbrales para cada uno de los sensores. Ejemplo, el rango que consideramos para la Tª está entre 10ºC como mínimo y 30ºC como máximo.
#define minTemperature 10
#define maxTemperature 30
#define minHumidity 5
#define maxHumidity 80
#define minLight 15
#define maxLight 95
#define maxWater 10

// Definimos las variables que utilizará el programa durante su ejecución
// Variable para interactuar con el display LCD
LiquidCrystal lcd(PIN1,PIN2,PIN3,PIN4,PIN5,PIN6);

// Variable de control para saber si existe alguna alarma
bool ExistAlarm;

// Variables para realizar cálculos
int value, normalizedValue;
```

```
// Variable para interactuar con el sensor DHT
SimpleDHT11 sensor;

// Variables de control de alarma de los diferentes sensores
bool AlarmFromTemperature;
bool AlarmFromHumidity;
bool AlarmFromLight;
bool AlarmFromPresence;
bool AlarmFromWater;

// Inicialización del programa
void setup()
{
	pinMode(PIN1, OUTPUT);
	pinMode(PIN2, OUTPUT);
	pinMode(PIN3, OUTPUT);
	pinMode(PIN4, OUTPUT);
	pinMode(PIN5, OUTPUT);
	pinMode(PIN6, OUTPUT);
	pinMode(BUZZER, OUTPUT);
	pinMode(LED1, OUTPUT);
	pinMode(LED2, OUTPUT);
	pinMode(PIRSENSOR, INPUT);
	pinMode(LDR, INPUT);
	pinMode(DHT, INPUT);
	pinMode(WATER, INPUT);

//Inicialización de la variable de la pantalla LCD
	lcd.begin(16, 2);
}

// Ejecución del programa
void loop()
{
	// Las variables se igualan todas a False para volver a comprobar los sensores
		ExistAlarm = false;
		AlarmFromTemperature = false;
		AlarmFromHumidity = false;
		AlarmFromLight = false;
		AlarmFromPresence = false;
		AlarmFromWater = false;

	// Comprobación sensor LDR
		value = analogRead(LDR);
		normalizedValue = map(value,0,1023,0,100);
		if(normalizedValue < minLight || normalizedValue > maxLight)
		{
```

```
                ExistAlarm = true;
                AlarmFromLight = true;
    }
```

// Comprobación sensor de agua

```
    if(analogRead(WATER)>maxWater)
    {
                ExistAlarm = true;
                AlarmFromWater = true;
    }
```

// Comprobación sensor de temperatura y humedad

```
    byte temperature = 0;
    byte humidity = 0;
    byte data[40] = {0};
    sensor.read(DHT, &temperature, &humidity, data);

    if(temperature < minTemperature || temperature > maxTemperature)
    {
                ExistAlarm = true;
                AlarmFromTemperature = true;
    }

    if(humidity < minHumidity || humidity > maxHumidity)
    {
                ExistAlarm = true;
                AlarmFromHumidity = true;
    }
```

// Comprobación sensor de presencia

```
    if(digitalRead(PIRSENSOR) == HIGH)
    {
                ExistAlarm = true;
                AlarmFromPresence = true;
    }
```

/* Comprobamos si ha existido alarma.

Si existe alarma (if), el display LCD muestra dicha alarma junto con el sensor o sensores que la han provocado.

Si no existe alarma (else), el display LCD mostrará el valor de todos los sensores */

```
    if(ExistAlarm)
    {
         lcd.clear();
                lcd.setCursor(0, 0);
                lcd.print("ALARMA");
                lcd.setCursor(0,1);
                if(AlarmFromTemperature)
                {
                        lcd.print("T ");
```

```
			}
			if(AlarmFromHumidity)
			{
				lcd.print("H ");
			}
			if(AlarmFromLight)
			{
				lcd.print("L ");
			}
			if(AlarmFromPresence)
			{
				lcd.print("P ");
			}
			if(AlarmFromWater)
			{
				lcd.print("W ");
			}

			digitalWrite(BUZZER, HIGH);
			digitalWrite(LED1, HIGH);
			digitalWrite(LED2, HIGH);
			delay(1000);
		}
		else
		{
			digitalWrite(BUZZER, LOW);
			digitalWrite(LED1, LOW);
			digitalWrite(LED2, LOW);

			lcd.clear();
			lcd.setCursor(0, 0);
			lcd.print("L:");
			lcd.print(normalizedValue);
			lcd.print("% ");
			lcd.print("H:");
			lcd.print(humidity);
			lcd.print("%");
			lcd.setCursor(0,1);
			lcd.print("T:");
			lcd.print(temperature);
			lcd.print("*C ");
			delay(1000);
		}
}
```

¡Enhorabuena! ¡Has llegado al final del aprendizaje! Para que seas consciente de todo lo que has aprendido en un fin de semana te hemos preparado un resumen con los hitos que has alcanzado:

- Conocimientos relacionados con la Robótica, Electrónica y Desarrollo de Software.
- Conocimiento del funcionamiento electrónico de dispositivos básicos: resistencias, LEDs, sensores, etc.
- Montaje de circuitos.
- Utilización del entorno de desarrollo de Arduino.
- Utilización de la placa Arduino.
- Utilización de las entradas y salidas de la placa de Arduino.
- Manejo de LEDs.
- Manejo de LED RGB.
- Manejo de Pulsadores.
- Manejo de Potenciómetros.
- Manejo de Zumbadores.
- Manejo de diferentes sensores.
- Manejo de pantalla LCD.
- Utilización del Monitor Serie de Arduino.
- Realización de programas para Arduino.
- Realización de proyectos complejos con lógica de funcionamiento mediante código fuente.
- Utilización de librerías de Arduino.

Este libro y todo lo que rodea a Time of Software es el resultado de muchos años dedicados a la docencia en el ámbito tecnológico. Primero con grupos de Educación Secundaria Obligatoria y Bachillerato y posteriormente mediante la docencia a formadores.

El trabajo de creación del método de aprendizaje, sintetización y ordenación de toda la información teórica relacionada con Arduino y elaboración de las diferentes prácticas plasmadas en el libro son responsabilidad de las personas directamente responsables de **Time of Software**, Alfredo Moreno y Sheila Córcoles, apasionados por el mundo de Arduino y por la docencia.

Queremos agradecer a nuestras familias, amigos y compañeros de trabajo el apoyo incondicional y las aportaciones que han realizado al método de aprendizaje de Arduino que hemos desarrollado, ¡gracias por ser nuestros conejillos de indias! Sin vosotros esto no hubiera sido posible.

Y por supuesto gracias a ti por comprar "Aprende Arduino en un fin de semana", esperamos que hayas conseguido el objetivo que te propusiste cuando compraste el libro. Habrás podido comprobar que ésto es sólo el principio, que Arduino es un mundo apasionante. No tengas dudas en ponerte en contacto con nosotros para contarnos qué tal te ha ido y cómo te va, **¡NO ESTÁS SOLO!**

Table of Contents